



INFORMATYKA ■ ZASTOSOWANIA

Leszek Rutkowski

Metody i techniki sztucznej inteligencji

SPIS TREŚCI

Przedmowa	IX
1. Wstęp	1
2. Wybrane zagadnienia sztucznej inteligencji	5
2.1. Wprowadzenie	5
2.2. Rys historyczny sztucznej inteligencji	5
2.3. Systemy ekspertowe	7
2.4. Robotyka	8
2.5. Przetwarzanie mowy i języka naturalnego	10
2.6. Heurystyki i strategie poszukiwań	12
2.7. Kognitywistyka	13
2.8. Inteligencja mrówek	14
2.9. Sztuczne życie	15
2.10. Boty	17
2.11. Perspektywy rozwoju sztucznej inteligencji	18
2.12. Uwagi	19
3. Metody reprezentacji wiedzy z wykorzystaniem zbiorów przybliżonych	20
3.1. Wprowadzenie	20
3.2. Pojęcia podstawowe	21
3.3. Aproxymacja zbioru	28
3.4. Aproxymacja rodziny zbiorów	36
3.5. Analiza tablic decyzyjnych	38
3.6. Zastosowanie programu LERS	45
3.7. Uwagi	50
4. Metody reprezentacji wiedzy z wykorzystaniem zbiorów rozmytych typu 1	52
4.1. Wprowadzenie	52
4.2. Podstawowe pojęcia i definicje teorii zbiorów rozmytych	52
4.3. Operacje na zbiorach rozmytych	63
4.4. Zasada rozszerzania	69
4.5. Liczby rozmyte	72
4.6. Normy trójkątne i negacje	79
4.7. Relacje rozmyte i ich właściwości	90
4.8. Przybliżone wnioskowanie	94
4.8.1. Podstawowe reguły wnioskowania w logice dwuwartościowej	94
4.8.2. Podstawowe reguły wnioskowania w logice rozmytej	95

4.8.3. Reguły wnioskowania dla modelu Mamdaniego	99
4.8.4. Reguły wnioskowania dla modelu logicznego	100
4.9. Rozmyte systemy wnioskujące	103
4.9.1. Baza reguł	104
4.9.2. Blok rozmywania	105
4.9.3. Blok wnioskowania	105
4.9.4. Blok wyostrzania	112
4.10. Zastosowania zbiorów rozmytych	114
4.10.1. Rozmyta metoda Delphi	114
4.10.2. Ważona rozmyta metoda Delphi	117
4.10.3. Rozmyta metoda PERT	118
4.10.4. Podejmowanie decyzji w otoczeniu rozmytym	121
4.11. Uwagi	131
5. Metody reprezentacji wiedzy z wykorzystaniem zbiorów rozmytych typu 2	132
5.1. Wprowadzenie	132
5.2. Podstawowe definicje	133
5.3. Ślad niepewności	136
5.4. Osadzone zbiory rozmyte	137
5.5. Podstawowe operacje na zbiorach rozmytych typu 2	139
5.6. Relacje rozmyte typu 2	144
5.7. Redukcja typu	146
5.8. Rozmyte systemy wnioskujące typu 2	151
5.8.1. Blok rozmywania	151
5.8.2. Baza reguł	153
5.8.3. Blok wnioskowania	153
5.9. Uwagi	158
6. Sieci neuronowe i algorytmy ich uczenia	159
6.1. Wprowadzenie	159
6.2. Neuron i jego modele	159
6.2.1. Budowa i działanie pojedynczego neuronu	159
6.2.2. Perceptron	161
6.2.3. Model Adaline	167
6.2.4. Model neuronu sigmoidalnego	172
6.2.5. Model neuronu Hebb'a	177
6.3. Sieci jednokierunkowe wielowarstwowe	178
6.3.1. Budowa i działanie sieci	178
6.3.2. Algorytm wstecznej propagacji błędów	180
6.3.3. Algorytm wstecznej propagacji błędów z członem momentum	187
6.3.4. Algorytm zmiennej metryki	188
6.3.5. Algorytm Levenberga–Marquardta	189
6.3.6. Rekurencyjna metoda najmniejszych kwadratów	191
6.3.7. Dobór architektury sieci	193
6.4. Sieci rekurencyjne	199
6.4.1. Sieć Hopfielda	200
6.4.2. Sieć Hamminga	203
6.4.3. Sieci wielowarstwowe ze sprzężeniem zwrotnym	205
6.4.4. Sieć BAM	205

6.5. Sieci samoorganizujące z konkurencją	206
6.5.1. Sieci typu WTA	207
6.5.2. Sieci typu WTM	212
6.6. Sieci typu ART	216
6.7. Sieci radialne	220
6.8. Probabilistyczne sieci neuronowe	225
6.9. Uwagi	227
7. Algorytmy ewolucyjne	229
7.1. Wprowadzenie	229
7.2. Problemy optymalizacji a algorytmy ewolucyjne	230
7.3. Rodzaje algorytmów zaliczanych do algorytmów ewolucyjnych	231
7.3.1. Klasyczny algorytm genetyczny	232
7.3.2. Strategie ewolucyjne	250
7.3.3. Programowanie ewolucyjne	266
7.3.4. Programowanie genetyczne	266
7.4. Zaawansowane techniki w algorytmach ewolucyjnych	269
7.4.1. Eksploracja i eksploatacja	269
7.4.2. Metody selekcji	269
7.4.3. Skalowanie funkcji przystosowania	272
7.4.4. Szczególne procedury reprodukcji	273
7.4.5. Metody kodowania	274
7.4.6. Rodzaje krzyżowań	277
7.4.7. Rodzaje mutacji	278
7.4.8. Inwersja	279
7.5. Algorytmy ewolucyjne w projektowaniu sieci neuronowych	280
7.5.1. Algorytmy ewolucyjne do uczenia wag sieci neuronowych	280
7.5.2. Algorytmy ewolucyjne do określania topologii sieci neuronowej	283
7.5.3. Algorytmy ewolucyjne do uczenia wag i określania topologii sieci neuronowej	285
7.6. Algorytmy ewolucyjne a systemy rozmyte	288
7.6.1. Systemy rozmyte do kontroli ewolucji	288
7.6.2. Ewolucja systemów rozmytych	290
7.7. Uwagi	297
8. Metody grupowania danych	300
8.1. Wprowadzenie	300
8.2. Podziały ostre i rozmyte	301
8.3. Miary odległości	305
8.4. Algorytm HCM	307
8.5. Algorytm FCM	308
8.6. Algorytm PCM	310
8.7. Algorytm Gustafsona–Kessela	311
8.8. Algorytm FMLE	313
8.9. Kryteria jakości grupowania	314
8.10. Ilustracja działania algorytmów grupowania danych	315
8.11. Uwagi	317
9. Systemy neuronowo-rozmyte typu Mamdaniego, logicznego i Takagi–Sugeno	318
9.1. Wprowadzenie	318

9.2.	Opis wykorzystywanych problemów symulacyjnych	319
9.2.1.	Polimeryzacja	319
9.2.2.	Modelowanie statycznej funkcji nieliniowej	320
9.2.3.	Modelowanie nieliniowego obiektu dynamicznego	320
9.2.4.	Modelowanie smaku ryżu	320
9.2.5.	Rozpoznawanie gatunku wina	321
9.2.6.	Klasyfikacja kwiatu irysa	321
9.3.	Systemy neuronowo-rozmyte typu Mamdaniego	322
9.3.1.	Systemy typu A	322
9.3.2.	Systemy typu B	324
9.3.3.	Systemy typu Mamdaniego w zadaniach modelowania	325
9.4.	Systemy neuronowo-rozmyte typu logicznego	335
9.4.1.	Systemy typu M1	336
9.4.2.	Systemy typu M2	342
9.4.3.	Systemy typu M3	347
9.5.	Systemy neuronowo-rozmyte typu Takagi-Sugeno	352
9.5.1.	Systemy typu M1	353
9.5.2.	Systemy typu M2	355
9.5.3.	Systemy typu M3	356
9.6.	Algorytm uczenia systemów neuronowo-rozmytych	358
9.7.	Ocena działania systemów neuronowo-rozmytych	373
9.7.1.	Kryteria oceny modeli z uwzględnieniem ich złożoności	374
9.7.2.	Metoda linii izokryterialnych	376
9.8.	Uwagi	382
10.	Elastyczne systemy neuronowo-rozmyte	383
10.1.	Wprowadzenie	383
10.2.	Miękkie normy trójkątne	383
10.3.	Parametryzowane normy trójkątne	386
10.4.	Przełączane normy trójkątne	389
10.5.	Systemy elastyczne	394
10.6.	Algorytmy uczenia	395
10.6.1.	Operatory podstawowe	401
10.6.2.	Funkcje przynależności	402
10.6.3.	Funkcje zakresowe	403
10.6.4.	<i>H</i> -funkcje	404
10.7.	Przykłady symulacyjne	407
10.7.1.	Polimeryzacja	408
10.7.2.	Modelowanie smaku ryżu	410
10.7.3.	Klasyfikacja kwiatu irysa	412
10.7.4.	Rozpoznawanie gatunku wina	414
10.8.	Uwagi	416
Literatura		417
Skorowidz		433

PRZEDMOWA

Pojawienie się niniejszej książki na polskim rynku wydawniczym jest wydarzeniem szczególnym. Ta cenna pozycja wypełnia poważną lukę w krajowej literaturze naukowo-technicznej. Jednocześnie przybliża ona czytelnikowi najnowsze dokonania w szybko rozwijającej się gałęzi wiedzy, jaką od kilkunastu lat pozostaje inteligencja obliczeniowa. Dziedzina ta, będąca przedmiotem niniejszej książki, jest jedną z tych ważnych dziedzin nauki, które pozwalają przetwarzanie informacje zawarte w danych i nadawać im zaprogramowaną przez użytkownika sensowną interpretację.

Ostatnie dekady upływają pod znakiem burzliwego rozwoju technik komputerowych i związanych z nimi metod obliczeniowych. Jednocześnie z ich pojawiением się i szybkim postępem, rozwijają się te gałęzie nauk teoretycznych i stosowanych, które pozwalają użytkownikowi na pełniejsze wykorzystanie nowo powstającego potencjału obliczeniowego i wydobywania wiedzy z coraz większej obfitości danych. Rozwój inteligencji obliczeniowej jest więc ściśle związany ze wzrostem ilości dostępnych danych, jak też i mocy ich przetwarzania, czynników wzajemnie się wspomagających. Bez nich rozwój tej dziedziny byłby prawie niemożliwy, a jej zastosowania — praktycznie marginalne. Dlatego właśnie techniki te rozwinęły się szczególnie w ostatnich kilkunastu latach.

Rozwój systemów inteligencji obliczeniowej był inspirowany przez możliwe do zaobserwowania i naśladowania aspekty inteligentnej aktywności człowieka i natury. Natura, podejmując inteligentne działania, przetwarza dane równolegle, regulując i korygując te działania poprzez mechanizmy sprzężenia zwrotnego. W układzie takim funkcjonują uczące się sieci neuronowe. Jako inny przykład mogą służyć tutaj optymalizacyjne metody algorytmiczne, wzorowane na procesach doboru naturalnego, czy też systemy logiki rozmytej, odzwierciedlające nieostrość, rozmytość, subiektywizm czy relatywizm ocen człowieka.

Inteligencja obliczeniowa jest nową gałęzią nauki, a jej rozwój datuje się od lat 60. ubiegłego wieku, kiedy to zostały opracowane pierwsze algorytmy uczących się maszyn — prekursorów dzisiejszych sieci neuronowych. Następnie, w latach 70. zostały stworzone podwaliny teorii zbiorów i logiki rozmytej. W tym wczesnym okresie rozwoju inteligencji obliczeniowej wprowadzono algorytmy genetyczne i ewolucyjne. W latach 80. stworzono też podstawy reprezentacji wiedzy z wykorzystaniem zbiorów przybliżonych. W ostatnich latach opracowano też wiele technik hybrydowych, łączących systemy uczące się z ewolucyjnymi i rozmytymi.

Opracowane teorie inteligencji obliczeniowej szybko znalazły liczne zastosowania w wielu dziedzinach inżynierii, analizy danych, prognozowania, w biomedycynie i innych. Stosuje się je w obróbce i rozpoznawaniu obrazów oraz dźwięków, przetwarzaniu sygnałów, wizualizacji wielowymiarowych danych, sterowaniu obiektami, analizie danych leksykograficznych, systemach wnioskujących w bankowości, systemach diagnostycznych, ekspertowych i w wielu innych praktycznych wdrożeniach.

Istotą systemów opartych na inteligencji obliczeniowej jest przetwarzanie i interpretacja danych o bardzo różnorakim charakterze. Mogą to być dane numeryczne, symboliczne (m.in. językowe o różnym stopniu precyzji), binarne, logiczne lub na przykład odczytane wprost na ekranie kamery niezakodowane obrazy. Dane te mogą być sformułowane jako liczby, czyli pojedyncze elementy wektorów, jako wektory lub tablice liczb bądź jako ciągi elementów lub złożonych z nich tablic. Mogą się też one składać z uporządkowanych sekwencji elementów lub tablic i zawierać elementy opisane w bardzo nieprecyzyjny, a nawet subiektywny sposób.

Wspólną cechą systemów inteligencji obliczeniowej jest to, że przetwarzają one informacje w przypadkach trudnych do przedstawienia w postaci algorytmów i czynią to w powiązaniu z symboliczną reprezentacją wiedzy. Mogą to być relacje dotyczące jakiegoś obiektu znanego tylko na podstawie skończonej liczby pomiarów stanu wyjścia i wejścia (pobudzenia). Mogą to być również dane wiążące najbardziej prawdopodobną diagnozę z szeregiem zaobserwowanych symptomów, często opisowych. W innych przypadkach mogą to być dane charakteryzujące zbiory względem ich pewnych specjalnych cech, które są dla użytkownika początkowo nieuchwytne aż do momentu ich wydobycia z danych i określenia jako cech dominujących. Systemy te mają zdolność odtwarzania zachowań zaobserwowanych w ciągach uczących, potrafią formułować reguły wnioskowania i generalizować wiedzę w sytuacjach, kiedy oczekuje się od nich predykcji bądź zaklasyfikowania obiektu do jednej z zaobserwowanych wcześniej kategorii.

Niniejsza książka nie tylko jest cenną pozycją na polskim rynku wydawniczym, lecz stanowi również udaną próbę syntezy metod inteligencji obliczeniowej w skali literatury światowej. Szczególną zaletą książki jest to, że zawiera ona szereg przykładów i ilustracji opisywanych metod, przez co stwarza dogodne możliwości zaprogramowania prezentowanych algorytmów. Książka ta zasługuje na polecenie jej inżynierom różnych specjalności, fizykom, matematykom, informatykom, ekonomistom oraz studentom wyższych lat studiów na tych lub pokrewnych kierunkach. Powinna ona przysporzyć wiele satysfakcji zarówno autorowi z faktu jej opublikowania, jak też i zastępującym czytelników, którzy wykorzystają opisane w niej techniki do rozwiązywania interesujących ich praktycznych zagadnień.

Na zakończenie warto dodać, że do rozwoju tej gałęzi nauk informatycznych przyczyniło się szczególnie wielu Polaków. Jest na tej liście wielu autorów, badaczy i pracowników naukowych, którzy dokonali istotnego wkładu w rozwój inteligencji obliczeniowej w Polsce bądź poza jej granicami. Można tu wymienić takie zasłużone osoby, jak Józef Arabas, Mariusz Barski, Michał Biały, Leonard Bolc, Zdzisław Bubnicki, Andrzej Cader, Andrzej Cichocki, Krzysztof Cios, Włodzisław Duch, Jerzy Grzymała-Busse, Wojciech Jędruch, Janusz Kacprzyk, Tadeusz Kaczorek, Józef Korbicz, Jacek Koronacki, Robert A. Kosiński, Witold Kosiński, Bohdan Macukow, Jacek Mańdziuk, Zbigniew Mi-

chalewicz, Andrzej Obuchowicz, Stanisław Osowski, Andrzej Pacut, Zdzisław Pawlak, Witold Pedrycz, Andrzej Piega, Maciej Piliński, Zbigniew Raś, Danuta Rutkowska, Języ Seidler, Andrzej Skowron, Roman Słowiński, Janusz Starzyk, Ryszard Tadeusiewicz, Dariusz Uciński, Andrzej Zadrożny i wiele innych, których z braku miejsca na tej liście nie wymieniono.

18 lipca 2005 r.

Jacek M. Żurada

Członek Zagraniczny Polskiej Akademii Nauk

Prezydent Stowarzyszenia Inteligencji Obliczeniowej IEEE

Computational Intelligence Society, Institute of Electrical and Electronic Engineers, Inc.

New York, USA

Wstęp

Początków sztucznej inteligencji można się doszukiwać w odległych wiekach, nawet u starożytnych filozofów, w szczególności jeśli rozważamy właśnie filozoficzne aspekty tej dziedziny nauki. Mniej odległa czasowo jest pierwsza połowa XIX wieku, kiedy to profesor Uniwersytetu w Cambridge Charles Babbage wpadł na pomysł tzw. „maszyny analitycznej”, realizującej nie tylko działania arytmetyczne określonego typu, ale również mogącej wykonywać działania zgodnie z wcześniej przygotowanymi instrukcjami. Istotną rolę w tym projekcie odgrywały karty dziurkowane, które stały lat później okazały się bardzo ważnym elementem komunikacji człowieka z komputerem. W roku 1950 Alan Turing zaproponował test mający stwierdzić, czy dany program jest inteligentny. Niedługo później powstają konkretne prace i są realizowane projekty badawcze dotyczące rozumienia języka naturalnego i rozwiązywania złożonych problemów. Ambicją uczonych stało się stworzenie uniwersalnego systemu, o nazwie „General Problem Solver”, mającego rozwiązywać problemy z różnych dziedzin. Projekt taki zakończył się niepowodzeniem, ale w toku jego realizacji badacze mieli okazję zgłębić złożoność problematyki sztucznej inteligencji. Lata 60. i 70. ubiegłego wieku charakteryzują się całkowitą dominacją tzw. podejścia symbolicznego do rozwiązywania różnych zagadnień sztucznej inteligencji. Tak więc stosowano metody indukcji drzew decyzyjnych, logiki predykatów i w pewnym stopniu klasyczne metody probabilistyczne, które jednak nabraly większego znaczenia w późniejszym okresie, z chwilą rozwoju sieci bayesowskich. Cechą znamienną tamtego okresu było odzegnywanie się od zastosowania obliczeń numerycznych do rozwiązywania problemów sztucznej inteligencji. Punktem zwrotnym w rozwoju sztucznej inteligencji było opublikowanie w 1986 roku książki, w której Rumelhart i McClelland podali sposób uczenia wielowarstwowych sieci neuronowych, co umożliwiło rozwiązywanie problemów, np. klasyfikacji, z którymi tradycyjne metody nie potrafiły sobie poradzić. Na początku lat dziewięćdziesiątych ideę uczenia sieci neuronowych zaadaptowano do uczenia systemów rozmytych. W ten sposób powstały struktury neuronowo-rozmyte, a ponadto zaproponowano różne inne kombinacje sieci neuronowych, systemów rozmytych i algorytmów ewolucyjnych. Dzisiaj mamy do czynienia z oddzielną gałęzią nauki określonej w literaturze angielskojęzycznej terminem Computational Intelligence, co możemy przetłumaczyć na język polski jako inteligencja obliczeniowa. Pod tym terminem rozumiemy rozwiązywanie różnych problemów sztucznej inteligencji z wykorzystaniem komputerów wykonujących obliczenia numeryczne.

Obliczenia te związane są z zastosowaniem następujących technik:

- a) sieci neuronowe [242, 270],
- b) logika rozmyta [94, 265],
- c) algorytmy ewolucyjne [57, 136],
- d) zbiory przybliżone [161, 163],
- e) zmienne niepewne [18, 19],
- f) metody probabilistyczne [1, 157].

Powyżej zacytowano jedynie wybrane prace lub monografie przedstawiające te techniki, zaliczane do technik leżących w obrębie metod „miękkich obliczeń” (ang. *soft techniques* [1, 108]). Należy podkreślić, że przedmiotem zainteresowań inteligencji obliczeniowej są nie tylko pojedyncze techniki, ale również ich rozmaite kombinacje [104]. W skali międzynarodowej działa towarzystwo o nazwie IEEE Computational Intelligence Society, które organizuje liczne konferencje w zakresie inteligencji obliczeniowej, a ponadto wydaje trzy prestiżowe czasopisma w tej dziedzinie, tzn.: *IEEE Transactions on Neural Networks*, *IEEE Transactions on Fuzzy Systems* oraz *IEEE Transactions on Evolutionary Computation*. W Polsce działa polska sekcja tego towarzystwa. Metody sztucznej inteligencji oraz inteligencji obliczeniowej leżą także w obszarze zainteresowań Polskiego Towarzystwa Sieci Neuronowych, które w cyklu dwuletnim organizuje konferencje o nazwie „Artificial Intelligence and Soft Computing”. Celem tych konferencji jest integracja badaczy reprezentujących zarówno tradycyjne podejście do metod sztucznej inteligencji, jak i stosujących metody inteligencji obliczeniowej.

Tematyka tej książki dotyczy różnych technik inteligencji obliczeniowej, zarówno pojedynczych, jak i tworzących metody hybrydowe. Techniki te dzisiaj są powszechnie stosowane do klasycznych zagadnień sztucznej inteligencji, np. do przetwarzania mowy i języka naturalnego, budowy systemów ekspertowych i robotów, wyszukiwania informacji oraz uczenia się maszyn. Poniżej wyszczególniono główne wątki tej książki.

W rozdziale 2 krótko przedstawiono wybrane zagadnienia sztucznej inteligencji, rozpoczynając od historycznego już dzisiaj testu Turinga oraz zagadnienia „Chińskiego pokoju”. Rozdział ten zawiera wstępne informacje o systemach ekspertowych, robotyce, problemach przetwarzania mowy i języka naturalnego oraz metodach heurystycznych. W drugiej części rozdziału zwrócono uwagę na ważność kognitywistyki, czyli nauki, która próbuje zrozumieć naturę umysłu. Później przybliżono czytelnikowi kolejno zagadnienia inteligencji mrówek oraz algorytmów mrówkowych, dziedziny nauki o nazwie „sztuczne życie”, jak również inteligentnych programów komputerowych, znanych pod nazwą boty. W zakończeniu rozdziału podano opinie znanych naukowców na temat perspektyw sztucznej inteligencji oraz sformułowano wnioski odzwierciedlające poglądy autora książki na ten temat.

Następne trzy rozdziały dotyczą metod reprezentacji wiedzy z wykorzystaniem różnych technik, a mianowicie zbiorów przybliżonych, zbiorów rozmytych typu 1 oraz zbiorów rozmytych typu 2.

W rozdziale 3 podano podstawowe informacje na temat zbiorów przybliżonych. Omówiono zagadnienie aproksymacji zbioru oraz rodziny zbiorów. W drugiej części rozdziału przedstawiono problematykę tablic decyzyjnych, a następnie zastosowano program LERS do generowania bazy reguł. Rozdział ten, podobnie jak dwa następne,

jest bogato ilustrowany przykładami ułatwiającymi czytelnikowi zrozumienie różnych definicji.

W rozdziale 4 przedstawiono podstawowe pojęcia i definicje teorii zbiorów rozmytych. Następnie omówiono zagadnienie przybliżonego wnioskowania, tzn. wnioskowania na podstawie rozmytych przesłanek. Ponadto zapoznano czytelnika z metodą konstrukcji rozmytych systemów wnioskujących. W drugiej części rozdziału podano szereg przykładów zastosowań zbiorów rozmytych w zagadnieniach prognozowania, planowania oraz podejmowania decyzji.

W rozdziale 5 przedstawiono podstawowe definicje dotyczące zbiorów rozmytych typu 2, omówiono operacje na tych zbiorach, a następnie relacje rozmyte typu 2. Dużo uwagi poświęcono metodzie redukcji typu, czyli sposobowi transformacji zbiorów rozmytych typu 2 w zbiory rozmyte typu 1. W ostatniej części rozdziału przybliżono czytelnikowi zagadnienie projektowania rozmytych systemów wnioskujących typu 2.

Rozdział 6 jest poświęcony sztucznym sieciom neuronowym. W rozdziale tym najpierw przedstawiono różne modele matematyczne pojedynczego neuronu. Następnie szczegółowo omówiono budowę i działanie wielowarstwowych sieci neuronowych. Podano szereg algorytmów uczenia tych sieci oraz zwróciły uwagę na zagadnienie doboru ich architektury. W kolejnych punktach zapoznano czytelnika z ideą sieci neuronowych ze sprzężeniem zwrotnym. Omówiono budowę i działanie sieci Hopfielda, Hamminga, Elmana, RTRN oraz BAM. W drugiej części rozdziału przedstawiono zagadnienie sieci samoorganizujących z konkurencją, sieci typu ART, sieci radialnych oraz probabilistycznych sieci neuronowych.

W rozdziale 7 omówiono rodzinę algorytmów ewolucyjnych, a w szczególności klasyczny algorytm genetyczny, strategie ewolucyjne oraz programowanie genetyczne. Przedstawiono też zaawansowane techniki w algorytmach ewolucyjnych. W drugiej części rozdziału podano różne sposoby powiązania technik ewolucyjnych z sieciami neuronowymi oraz systemami rozmytymi.

W rozdziale 8 przedstawiono różne sposoby podziału danych oraz algorytmy automatycznego ich grupowania. Zdefiniowano podziały ostre, rozmyte i posybilistyczne. Następnie podano stosowane w metodach grupowania miary odległości, po czym omówiono najbardziej popularne algorytmy grupowania danych, tzn. algorytm HCM, algorytm FCM, algorytm PCM, algorytm Gustafsona–Kessela i algorytm FMLE. Rozdział kończy przedstawienie znanych metod oceny jakości grupowania danych.

W rozdziale 9 wyprowadzono różne struktury neuronowo-rozmyte. Struktury te są wielowarstwową (sieciową) reprezentacją klasycznego systemu rozmytego. Do ich konstrukcji zastosowano wnioskowanie typu Mamdaniego oraz typu logicznego. Ponadto rozważano tzw. schemat Takagi–Sugeno, w którym następni reguł nie mają charakteru rozmytego, ale są funkcjami zmiennych wejściowych. Cechą charakterystyczną wszystkich struktur jest możliwość wprowadzenia wag odzwierciedlających ważność zarówno poszczególnych wartości lingwistycznych w poprzednikach rozmytych reguł, jak i wag odzwierciedlających ważność całych reguł. Do konstrukcji tych struktur wykorzystano podaną w rozdziale 4 koncepcję ważonych norm trójkątnych. Normy te nie spełniają warunków brzegowych klasycznej t -normy oraz t -konormy, podobnie jak powszechnie stosowana reguła wnioskowania typu Mamdaniego nie spełnia warunków implikacji lo-

gicznej. W rozdziale tym pokazano, że zastosowanie ważonych norm trójkątnych prowadzi do konstrukcji struktur neuronowo-rozmytych charakteryzujących się bardzo małym błędem działania systemu. W drugiej części rozdziału przedstawiono algorytmy uczenia wszystkich struktur, a następnie rozwiązano zagadnienie zaprojektowania systemów neuronowo-rozmytych, które charakteryzują się osiągnięciem kompromisu między błędem działania systemu a liczbą parametrów opisujących ten system.

W rozdziale 10 przedstawiono koncepcje tzw. elastycznych systemów neuronowo-rozmytych. Ich cechą charakterystyczną jest możliwość znalezienia sposobu wnioskowania (typu Mamdaniego lub logicznego) w wyniku procesu uczenia. Realizacja takich systemów będzie możliwa dzięki specjalnie skonstruowanym i pokazanym w tym rozdziale przełączanym normom trójkątnym. Ponadto, do konstrukcji systemów neuronowo-rozmytych wprowadzono koncepcję miękkich norm trójkątnych, parametryzowanych norm trójkątnych oraz, już wcześniej stosowanych w rozdziale 9, wag opisujących ważność poszczególnych reguł oraz przesłanek w tych regułach.

Niektóre wyniki przedstawione w książce są rezultatem badań przeprowadzonych w ramach realizacji projektu badawczego 3 T11C 048 27 finansowanego przez Komitet Badań Naukowych.

Książka jest wynikiem między innymi wykładów prowadzonych przez autora w ostatnich latach dla magistrantów Politechniki Częstochowskiej oraz Wyższej Szkoły Humanistyczno-Ekonomicznej w Łodzi, jak również doktorantów Instytutu Badań Systemowych Polskiej Akademii Nauk.

Książka jest też wynikiem współpracy z Kolegami z Katedry Inżynierii Komputerowej Politechniki Częstochowskiej, którzy tajniki inteligencji obliczeniowej zdobywali jeszcze jako moi studenci na czwartym roku studiów. Tak więc pragnę serdecznie podziękować za okazaną mi pomoc dr. inż. Krzysztofowi Cpałce, dr. inż. Robertowi Nowickiemu, dr. inż. Rafałowi Schererowi oraz dr. inż. Januszowi Starczewskiemu. Ponadto dziękuję przedstawicielom nieco młodszego pokolenia Katedry Inżynierii Komputerowej, a więc mgr. inż. Marcinowi Gabryelowi, mgr. inż. Marcinowi Korytkowskiemu oraz dr inż. Agacie Pokropińskiej. Trud przygotowania części rysunków podjęła Pani mgr Renata Marciniak, której autor również serdecznie dziękuje.

Serdeczne podziękowania należą się recenzentom, Panu Profesorowi Januszowi Kacprzykowi oraz Panu Profesorowi Ryszardowi Tadeusiewiczowi, którzy pierwsi zapoznali się z ideą przygotowania tej książki oraz wnieśli cenne uwagi.

Wybrane zagadnienia sztucznej inteligencji

2.1. Wprowadzenie

Rozważając zagadnienia sztucznej inteligencji, powinniśmy mieć pewien punkt odniesienia. Takim punktem odniesienia może być definicja ludzkiej inteligencji. W literaturze spotyka się rozmaite definicje, ale większość z nich sprowadza się do stwierdzenia, że inteligencja jest umiejętnością przystosowywania się do nowych zadań i warunków życia albo sposobem, w jaki człowiek przetwarza informacje i rozwiązuje problemy. Inteligencja to również umiejętność kojarzenia oraz rozumienia. Wpływ na nią mają zarówno czynniki dziedziczne, jak i wychowanie. Najważniejsze procesy i funkcje składające się na ludzką inteligencję to uczenie się i wykorzystywanie wiedzy, zdolność uogólniania, percepcja i zdolności poznawcze, np. zdolność rozpoznawania danego obiektu w dowolnym kontekście. Ponadto możemy wymienić takie elementy, jak zapamiętywanie, stawianie i realizacja celów, umiejętność współpracy, formułowanie wniosków, zdolność analizy, tworzenie oraz myślenie koncepcyjne i abstrakcyjne. Z inteligencją związane są też takie czynniki, jak samoświadomość, emocjonalne i irracjonalne stany człowieka.

Stworzone przez człowieka tzw. intelligentne maszyny można zaprogramować tak, aby jedynie w wąskim zakresie imitowały kilka wyżej wymienionych elementów składających się na ludzką inteligencję. Zatem czeka nas jeszcze długa droga do zrozumienia działania mózgu i zbudowania jego sztucznego odpowiednika. W tym rozdziale krótko przedstawimy wybrane zagadnienia sztucznej inteligencji, rozpoczynając od historycznego już dzisiaj testu Turinga i zagadnienia „Chińskiego pokoju”.

2.2. Rys historyczny sztucznej inteligencji

Sztuczna inteligencja (SI) to termin, który budzi wielkie zainteresowanie, a także wiele kontrowersji. Określenie to zaproponował po raz pierwszy John McCarthy w 1956 roku, organizując konferencję w Dartmouth College na temat intelligentnych maszyn. Do zagadnień SI należy między innymi poszukiwanie metod rozwiązywania problemów. Przykładem mogą być poszukiwanie algorytmów do gry w szachy. Rozumowanie logiczne to drugie z wielu zagadnień SI. Polega ono na zbudowaniu algorytmu naśladującego sposób wnioskowania, jaki zachodzi w mózgu. Kolejnym przedmiotem badań SI jest przetwarzanie języka naturalnego, a co za tym idzie, automatyczne tłumaczenie zdań między różnymi językami, wydawanie poleceń słownych maszynom, a także wydoby-

wanie informacji ze zdań mówionych i budowanie z nich baz wiedzy. Badacze SI stają przed wyzwaniem stworzenia programów, które uczą się na podstawie analogii i w ten sposób same potrafią się udoskonalać. Przewidywanie i prognozowanie wyników oraz planowanie to także domeny sztucznej inteligencji. Istnieje duża grupa filozofów zastanawiających się nad problemem świadomości intelligentnego komputera. Naukowcy próbują także zgłębiać procesy percepcji, tzn. wizji, doryku i słuchu, a co za tym idzie, zbudować elektroniczne odpowiedniki tych narządów i zastosować je w robotyce.

W literaturze przedstawiono rozmaite definicje sztucznej inteligencji:

a) Sztuczna inteligencja jest nauką o maszynach realizujących zadania, które wymagają inteligencji, gdy są wykonywane przez człowieka (M. Minsky).

b) Sztuczna inteligencja stanowi dziedzinę informatyki dotyczącą metod i technik wnioskowania symbolicznego przez komputer oraz symbolicznej reprezentacji wiedzy stosowanej podczas takiego wnioskowania (E. Feigenbaum).

c) Sztuczna inteligencja obejmuje rozwiązywanie problemów sposobami wzorowanymi na naturalnych działaniach i procesach poznawczych człowieka za pomocą symulujących je programów komputerowych (R.J. Schalkoff).

Mimo, że SI jest uznawana za dziedzinę informatyki, to budzi zainteresowanie wielu uczonych innych gałęzi wiedzy, np. filozofów, psychologów, lekarzy i matematyków. Można więc śmiało stwierdzić, iż jest to nauka interdyscyplinarna, która dąży do zbadania ludzkiej inteligencji i zimplementowania jej w maszynach. Znając definicję SI, możemy zadać pytanie: kiedy nasz program lub maszyna jest intelligentna? Na pytanie to próbował odpowiedzieć w 1950 roku angielski matematyk Alan Turing. Jest on pomysłodawcą, tzw. „Testu Turinga” mającego rozstrzygnąć, czy program jest intelligentny. Idea tego testu polega na tym, że człowiek za pomocą klawiatury i monitora zadaje te same pytania komputerowi i innej osobie. Jeżeli zadający pytania nie potrafi odróżnić odpowiedzi komputera od odpowiedzi człowieka, to możemy stwierdzić, iż komputer (program) jest intelligentny. Znanym krytykiem pomysłu Turinga był amerykański filozof John Searle. Twierdził on, że komputery nie mogą być intelligentne, bo chociaż posługują się symbolami według pewnych zasad, to nie rozumieją ich znaczenia. Na poparcie tej tezy filozof wymyślił przykład znany w literaturze pod nazwą „Chiński pokój”. Założymy, że mamy zamknięty pokój, w którym znajduje się Europejczyk nie znający języka chińskiego. Do pomieszczenia zostają mu podawane pojedyncze kartki zapisane chińskimi znakami i opowiadające pewną historię. Nasz bohater nie zna języka chińskiego, ale zauważa na półce książkę napisaną w znanym mu języku pod tytułem *Co zrobić, gdy ktoś wsunie pod drzwiami kartkę z chińskimi znakami*. W książce tej znajdują się instrukcje sporządzenia chińskich znaków skorelowanych z tymi, które otrzymał. Na każde pytanie Europejczyk przygotowuje odpowiedź zgodnie z zasadami podanymi w podręczniku. Searl stwierdza, że człowiek zamknięty w pokoju tak naprawdę nic nie rozumie z przekazywanych mu informacji, podobnie jak komputer wykonujący program. Zatem istnieje oczywista różnica między myśleniem a symulowaniem procesów myślowych. Według Searla, jeżeli nawet nie odróżnimy odpowiedzi maszyny od odpowiedzi człowieka, to nie oznacza, że maszyna jest intelligentna. Założymy jednak, że istnieje maszyna, która zdała test Turinga. Możemy, więc stwierdzić, że jest intelligentna i jako taka potrafi myśleć. W tej sytuacji Roger Penrose, autor książki *Nowy umysł cesarza*,

zastanawia się, czy wykorzystywanie jej do własnych potrzeb bez zwracania uwagi na jej pragnienia byłoby akceptowalne, czy raczej godne potępienia. Czy sprzedawanie takiej maszyny lub odłączenie jej od źródła zasilania, które w tym przypadku możemy traktować jako pokarm, byłoby moralne? Penrose w swej książce opisuje jedno z pierwszych urządzeń SI. Był to elektroniczny żółw zbudowany przez Greya w początku lat 50. ubiegłego wieku. Urządzenie to poruszało się po pokoju dzięki energii pobieranej z baterii. Gdy napięcie spadało poniżej pewnego progu, żółw sam szukał najbliższego kontaktu i ładował swe ogniwa. Zauważmy, że takie zachowanie jest analogiczne do poszukiwania i spożywania przez ludzi pokarmu. Penrose posuwa się dalej w swych rozważaniach i proponuje wprowadzenie pewnej miary „szczęścia żółwia” — wartości z przedziału np. -100 (żółw jest bardzo nieszczęśliwy) do +100 (skrajne szczęście). Założymy także, że nasze elektroniczne zwierzątko może uzupełniać swe zapasy energii, korzystając z energii słonecznej. Można przyjąć, iż żółw jest nieszczęśliwy, gdy znajduje się w ciemnym miejscu, gdzie słońce nie dociera (nic nie stoi na przeszkodzie, aby stwierdzić, iż nasze sztuczne zwierzątko jest głodne), natomiast ogromną „radość” sprawia mu kąpiel w promieniach słonecznych. Po takim opisie tego urządzenia niewiele osób mogłoby zamknąć żółwia w ciemnym pokoju, a przecież jest to tylko zwykła maszyna, taka jak np. komputer.

2.3. Systemy ekspertowe

Jak mówi jedna z wielu definicji, system ekspertowy jest „inteligentnym” programem komputerowym, stosującym wiedzę i procedury rozumowania (wnioskowania) w celu rozwiązywania problemów, które wymagają doświadczenia ludzkiego (eksperta), nabytego przez wieloletnią działalność w danej dziedzinie. Ogólna idea działania systemów ekspertowych polega na przeniesieniu wiedzy eksperta z danej dziedziny do bazy wiedzy, zaprojektowaniu maszyny wnioskującej na podstawie posiadanych informacji oraz dodaniu interfejsu użytkownika, służącego do komunikacji.

Pierwowzorem dla systemów ekspertowych był program DENDRAL, który opracowano w pierwszej połowie lat 60. ubiegłego wieku na Uniwersytecie Standforda. Jego zadaniem było obliczanie wszystkich możliwych konfiguracji danego zbioru atomów. Integralną częścią programu była baza wiedzy zawierająca prawa chemiczne i reguły, które przez dziesięciolecia wypracowano w laboratoriach chemicznych. Program DENDRAL stał się niezwykle pomocny w rozwiązywaniu zagadnień, dla których nie wypracowano metod analitycznych.

Na tym samym Uniwersytecie Standforda w latach 70. XX wieku powstały dwa systemy ekspertowe, które przeszły do historii jako wzorcowe rozwiązania w tym zakresie. System PROSPEKTOR został zaprojektowany, aby wspomagać geologów w określeniu rodzaju skały na podstawie zawartości różnych minerałów. Ułatwił poszukiwanie złóż surowców mineralnych i szacunek zasobów tych złóż. PROSPEKTOR był systemem konwersacyjnym, wykorzystującym reguły otrzymane od specjalistów. Modele poszczególnych typów złóż zawierały od kilkudziesięciu do kilkuset reguł stanowiących bazę wiedzy, która była odseparowana od mechanizmu wnioskowania. Zastosowanie systemu

PROSPEKTOR przyniosło spektakularne sukcesy w postaci odkrycia w stanie Washington (USA) bogatych złóż molibdenu. System MYCIN zaprojektowano w celu diagnozy chorób zakaźnych. Do systemu wprowadzano dane dotyczące pacjenta oraz wyniki testów laboratoryjnych. Rezultatem jego działania była diagnoza oraz zalecenia postępowania w przypadkach pewnych infekcji krwi. System ten wspomagał podejmowanie decyzji w sytuacji niekompletnych danych. W przypadku wątpliwości system podawał stopień pewności swojej diagnozy oraz alternatywne rozwiązania (diagnozy). Na bazie systemu MYCIN powstał system NEOMYCIN, który znalazł zastosowanie w szkoleniu lekarzy.

Warto wspomnieć o jednym z największych projektów w historii sztucznej inteligencji znany pod akronimem CYC (nazwa jest fragmentem słowa encyklopedia — ang. *encyclopedia*) i realizowanym w USA. System ten zawierał miliony reguł (docełlowo planowano 100 milionów reguł), co miało zapewnić nadzwyczajne możliwości „intelektualne” komputera zawierającego odpowiedni program.

Jak już wspomnieliśmy, podstawowe elementy systemu ekspertowego to baza wiedzy, maszyna wnioskująca i interfejs użytkownika. Na bazę wiedzy składa się zbiór faktów i reguł. Reguły są zdaniami logicznymi, które definiują pewne implikacje i prowadzą do stworzenia nowych faktów, co w efekcie pozwala rozwiązać dany problem. Maszyna wnioskująca jest modelem, który korzysta z bazy wiedzy. Moduł ten może wykorzystywać różne sposoby wnioskowania w celu rozwiązania problemu. Dużą popularnością cieszą się tzw. szkieletowe systemy ekspertowe, czyli programy komputerowe z zaprojektowaną maszyną wnioskującą i pustą bazą wiedzy. Częścią składową tych programów są specjalne edytory pozwalające na wpisanie reguł dotyczących problemu, który chce rozwiązać użytkownik. Problematyka konstruowania systemów ekspertowych należy do zagadnień, tzw. inżynierii wiedzy. W obszarze zainteresowań specjalistów zajmujących się inżynierią wiedzy są takie zagadnienia, jak pozyskiwanie wiedzy, jej strukturalizacja i przetwarzanie, projektowanie i wybór odpowiednich metod wnioskowania (maszyna wnioskująca) oraz projektowanie odpowiednich interfejsów między komputerem i użytkownikiem.

2.4. Robotyka

Pojęcie „robot” pojawiło się po raz pierwszy w 1920 roku, w sztuce „R.U.R.”, której autorem był czeski pisarz Karel Čapek. W dramacie została przedstawiona wizja niewłaściwego wykorzystania techniki przez ludzi. Historia toczy się wokół fabryki będącej wytwarznią robotów — niewolników, którzy mają zastąpić ludzi w wykonywaniu ciężkich obowiązków i trudnej pracy. Przemysł produkujący roboty jest rozwijany, a maszyny są unowocześniane oraz wyposażane w coraz większą inteligencję. Duży popyt pozwala na zwiększenie liczby budowanych robotów. Zaczęto je wykorzystywać do zadań wojskowych, w charakterze żołnierzy. Nadszedł czas, gdy roboty uzyskały liczebną przewagę nad swoimi twórcami — ludźmi. Koniec sztuki to bunt robotów i zagłada ludzkiego gatunku.

Dynamiczny rozwój robotów zainicjowały badania rozpoczęte w USA. W latach 50. zaczęły powstawać roboty przystosowane do pracy w fabrykach, które między in-

nymi składały samochody w fabryce General Motors. Zaczęto zajmować się pracami nad budową maszyn manipulacyjnych dla przemysłu nuklearnego i poszukiwań oceanograficznych. Obecnie roboty to cuda elektroniki, które potrafią się uczyć, a ceny ich często są większe od cen luksusowych samochodów. Wykorzystywane są praktycznie wszędzie. Wykonują prace od niepozornych i śmiesznych, takich jak podawanie kapci czy kawy, poprzez trudne i ciężkie dla człowieka prace w trudnych warunkach w przemyśle ciężkim, aż do skomplikowanych operacji chirurgicznych. W 2002 roku robot sterowany przez profesora Louisa Kavoussi z odległości tysiąca kilometrów wykonał operację chirurgiczną. Rola lekarzy nadzorujących przebieg pracy maszyny ograniczyła się do znieczulenia pacjenta. W ten sposób chory nie musi czekać na przyjazd lekarza, co zmniejsza znacznie koszty i czas zabiegu. Robot da Vinci firmy Intuitive Surgical naśladuje ruch rąk lekarza w czasie operacji, a jednocześnie eliminuje z nich drżenie. Ponadto wyświetla duży, powiększony obraz serca chorego. Ułatwia to przeprowadzenie zabiegu, gdyż lekarz dokładnie widzi operowany narząd. Precyza robotów powoduje, że znacznie zmniejsza się uszkodzenia tkanek chorego. Dzięki temu pacjent szybko może wrócić do zdrowia. Roboty zastępują często ludzi, gdy zachodzi konieczność wykonania niebezpiecznych prac, np. przy rozbijaniu bomb.

Od kilku lat japońska firma Honda pokazuje kolejne wersje robota ASIMO. Jego twórcy twierdzą, że robot zna dwa języki, angielski i japoński, oraz potrafi rozmawiać. Bez problemu porusza się po schodach i omija różne przeszkody. Ciekawym robotem jest też AIBO. Przybrał on formę metaliczno-srebrnego pieska, który potrafi bawić się piłką i np. siusiać. Ma jednak problemy z omijaniem przeszkód, nie umie się wspinać i podawać łapy. Jedna z wersji potrafiła rozpoznać 75 komend wydawanych głosem. Aby poznać działanie tej zabawki użytkownik ma do dyspozycji 150 stronnicową książkę.

W Polsce, pierwsze prace związane z budową maszyn manipulacyjnych podjęto na początku lat 70., natomiast pierwsze udane zastosowanie robotów miało miejsce w 1976 roku w Olkuskiej Fabryce Naczyń Emaliowanych. Zastosowano tam maszyny do natryskowego emaliowania wanien i zlewozmywaków. Inne roboty zostały zainstalowane w linii montażowej do punktowego zgrzewania nadwozia samochodu osobowego Polonez. Do najważniejszych instytucji i przedsiębiorstw, w których prowadzone były prace naukowo-badawcze, trzeba zaliczyć PIAP — Przemysłowy Instytut Automatyki i Pomiarów, IMP — Instytut Mechaniki Precyzyjnej w Warszawie i CBKO — Centrum Badawczo-Konstrukcyjne Obrabiarek w Pruszkowie. Od lat intensywnie jest rozwijana robotyzacja w Fabryce Samochodów Osobowych (FIAT-AUTO Poland), gdzie kilkudziesiąt robotów obsługuje stanowiska montażu i zgrzewania karoserii samochodowych.

Badacze stawiają sobie pytanie, na ile robot musi być inteligentny i na czym ma polegać jego inteligencja. Zaznaczają się dwa główne podejścia, zwane słabą i mocną hipotezą sztucznej inteligencji. *Słaba hipoteza* sztucznej inteligencji zakłada, że inteligentna maszyna potrafi symulować ludzki proces poznania, ale sama nie może doświadczać stanów psychicznych. Maszyna taka ma szanse przejść przez test Turinga. *Mocna hipoteza* sztucznej inteligencji prowadzi do konstrukcji maszyn zdolnych do osiągnięcia kognitywnych stanów psychicznych. Podejście to umożliwia skonstruowanie maszyny zdającej sobie sprawę z własnego istnienia, z prawdziwymi emocjami i świadomością. W wielu ośrodkach naukowych prowadzone są badania nad ludzkim mózgiem i całym

układem nerwowym człowieka. Poznanie zasad funkcjonujących w naturze umożliwi skonstruowanie „robota rozumnego”. Przykładem może być robot „Dynamic Brain”, którego twórcy (neurofizycy Stefan Schaal i Mitsuo Kawato) poszukiwali zasad uczenia się i samoorganizacji umożliwiających systemowi rozwinięcie w sobie inteligencji. Robot ten poprzez oglądanie filmu z kobietą wykonującą japoński taniec ludowy, sam nauczył się tańczyć. Autorzy projektu wykorzystują robota do badań działania ludzkiego mózgu i interakcji zachodzących między mózgiem a ciałem człowieka.

Inteligencję sześciolatnego dziecka miał osiągnąć robot o nazwie Cog, dzieło Rodneya Brooksa. Celem budowy było zbadanie zagadnień rozwoju robota, jego fizycznej personifikacji oraz połączenia możliwości sensoryczno-motorycznych i interakcji społecznych. Cog naśladował reakcje człowieka, potrafił skupić wzrok na przedmiotach i wyciągnąć ręce w ich stronę. W trakcie ruchu korygował swoje działania. Jego zdolności były rozwijane w kierunku umiejętności rozróżnienia przedmiotów i organizmów żywych. Przekonanie, że możliwości robota mogą być bardzo szerokie, wyraziła Cynthia Breazeal. Zbudowała ona robota „Kismet”, zdolnego do nauczenia się wielu zachowań. Powinien on umieć porozumiewać się z ludźmi, rozumieć ich emocje, a także wyrażać własne poprzez mimikę „twarz”. Perspektywy robotyki przybierają niewyobrażalne wcześniejszej rozmiary. Przykładowo w USA pod koniec XX wieku rozpoczęto wielki program badań w dziedzinie maszyn molekularnych. Jednym z podstawowych celów rozwoju nanotechnologii są małe roboty — nanoroboty, które mogą być pomocne we wspomaganiu układu immunologicznego, wykrywaniu bakterii, wirusów i komórek rakowych.

2.5. Przetwarzanie mowy i języka naturalnego

Oczywistą metodą porozumiewania się między ludźmi jest mowa. Komunikowanie się z komputerem lub innymi urządzeniami za pomocą języka mówionego może stać się duże ułatwienie w życiu ludzi głuchoniemych lub niepełnosprawnych. Twórca idei „Chińskiego pokoju” John Searl w sposób następujący odpowiedział na pytanie dotyczące najważniejszych dokonań SI: „Nie znam się na tyle dobrze na technologicznych osiągnięciach w tej materii, żeby podać konkretną odpowiedź. Jednakże zawsze fascynowały mnie dokonania w dziedzinie przetwarzania języka naturalnego. Uważam, że prace te zasługują na prawdziwe uznanie”. Badania w zakresie przetwarzania mowy i języka naturalnego obejmują następujące zagadnienia:

- a) syntezę mowy,
- b) rozumienie słowa mówionego,
- c) rozumienie języka naturalnego,
- d) tłumaczenie maszynowe.

Syntezę mowy można utożsamiać z próbą czytania książki przez komputer. Syntezą mowy ma liczne zastosowania, np. do nauki języków obcych lub do odczytywania informacji dla niewidomych. Badanie mowy nie jest łatwym problemem. Wynika to z faktu, że człowiek, wymawiając wyrazy, w odpowiedni sposób je intonuje. Aby dobrze wypowiedzieć dane zdanie, należy rozumieć jego sens, a komputer w tej kwestii (jak również

we wszystkich innych) nie jest świadomym. Ciekawym pomysłem było zastosowanie przez T.V. Ramana, kierownika zespołu informatyków firmy Adobe, różnych krojów czcionek w zależności od tego, jak komputer ma przeczytać dany tekst. Na przykład zdania napisane kursywą są czytane głośniej. Program, nad którym pracował, nosi nazwę ASTER (Audio System for Technical Readings). Stosowanie programu ASTER dla jego autora jest praktycznie koniecznością, gdyż stracił on wzrok w wieku 14 lat, ale używają go także jego koledzy z firmy Adobe, którym bardzo ułatwia codzienną pracę. Początkowo twórcy systemów naśladowujących ludzką mowę starali się tworzyć urządzenia wzorowane na ludzkim narządzie mowy. Niestety efekty działania maszyn i programów opartych na formantach nie były zadowalające i znacznie różniły się od ludzkiej mowy. Zrezygnowano więc z takiego podejścia i zaczęto w algorytmach wykorzystywać gotowe nagrane fragmenty mowy, które postanowiono w odpowiedni sposób sklejać. Okazuje się, iż był to bardzo dobry pomysł. Ideę tę stosowano w późniejszych rozmaitych modyfikacjach i udoskonaleniach tej metody.

Innym problemem, nad którym pracują naukowcy SI, jest rozumienie słowa mówionego (ang. *automatic speech recognition*). Rozwój tych badań umożliwia komunikację z komputerem, np. dyktowanie tekstów, wydawanie ustnych poleceń, czy też rozpoznanie (autoryzację) użytkownika po głosie. Jak wspominaliśmy, ludzie wymawiają słowa w różny sposób (intonacja, szybkość mówienia itp.), często niezgodnie z zasadami gramatyki. Przykładem działającego systemu może być system „Dragon Dictate”. W pierwszej fazie eksperymentów program ten wymagał kilkugodzinnego „dostrajania się” do sposobu mówienia osoby, która będzie dyktowała tekst. Obecnie system ten jest sprzedawany komercyjnie, podobnie jak i program konkurencyjny o nazwie Angora. W systemach tych wykorzystuje się bazy danych, w których umieszcza się wyrazy wraz z ich brzmieniem lub reprezentacją fonemową. Na zasadzie porównań system rozpoznaje słowo. Przykładem może być także np. wybieranie słowne w telefonach komórkowych, gdzie każde nazwisko ma swoją etykietę słowną zapisaną przez użytkownika.

Kolejnym zagadnieniem SI jest rozumienie języka naturalnego. Problem sprowadza się do wydobywania istotnych danych ze zdań zapisanych w postaci tekstu. Badacze tworzą systemy do wydobywania wiedzy ze zdań, a komputer powinien dokonać rozbioru zdania na części mowy. W ten sposób jest w stanie wydobyć z treści obiekty (rzeczowniki), ich cechy (przymiotniki) i związki między nimi. Dawniej systemy były przygotowywane do pracy dla specyficznych dziedzin nauki i zawierały dane z tej dziedziny w bazie wiedzy. Przykładem może być system Lunar, który odpowiadał na pytania dotyczące próbek skał przywiezionych z księżyca.

Ostatnim zagadnieniem SI związanym z językiem jest tłumaczenie maszynowe. Polega ono na tłumaczeniu tekstów między różnymi językami. Systemy tego typu są stosowane w siedzibie Unii Europejskiej. Należy zauważyć, że problemem może być różne znaczenie wyrazów w zależności od kontekstu. Czterdzieści lat temu w USA sformułowano raport, który zawierał stwierdzenie: „Nie udało się automatyczne tłumaczenie żadnego ogólnego tekstu naukowego i nie widać szans na szybki postęp w tej dziedzinie”. Obecnie na rynku są dostępne programy tłumaczące, pracujące na komputerach PC. Jednakże w dalszym ciągu pojawiają się problemy z tłumaczeniem tekstów zawierających zdania z wąskiej dziedziny, np. różne dokumenty techniczne. Przykładem może

być program Transcend. Działa on na komputerach osobistych i ma możliwość przetwarzania wielu tysięcy słów na minutę. Istnieją też systemy, które potrafią tłumaczyć słowa mówione (np. przez telefon) w czasie rzeczywistym. Wykorzystuje się do tego celu bardzo szybkie komputery, najczęściej wieloprocesorowe.

2.6. Heurystyki i strategie poszukiwań

Słowo „heurystyka” wywodzi się z greckiego słowa *heurisco*, co oznacza odkrywać, znajdować. Najprościej o heurystyce można powiedzieć jak o „twórczym rozwiązywaniu problemów”, zarówno logicznych, jak i matematycznych przez eksperyment, metodą prób i błędów bądź odwołaniem się do analogii. Metody heurystyczne znajdują zastosowania wszędzie tam, gdzie rozwiązanie problemu wymaga olbrzymich ilości obliczeń. Dzięki heurystyce możemy wyeliminować pewne obszary przeszukiwanej przestrzeni. W rezultacie znacznie zmniejszymy koszty obliczeniowe, a jednocześnie przyspieszymy znalezienie rozwiązania. W literaturze nie istnieją formalne dowody poprawności działania algorytmów heurystycznych, ale o skuteczności ich działania świadczą przeprowadzone symulacje. Szerokie zastosowanie znajdują między innymi w systemach ekspertowych, systemach wspomagania decyzji i badaniach operacyjnych. Pokonanie mistrza świata w szachach przez komputer stało się możliwe między innymi dzięki technikom heurystycznym, które pozwalały wykluczyć warianty nie rokujące sukcesu. Aby zrozumieć, czym jest heurystyka, zapoznajmy się ze znany w literaturze [70] przykładem. Przypuśćmy, że komuś upadło szkło kontaktowe. Oto kilka możliwości poszukiwań:

1. Szukanie ślepe — schylanie się i szukanie po omacku. Takie szukanie nie gwarantuje pozytywnego rezultatu.
2. Szukanie systematyczne — polega na rozszerzaniu przeszukiwanej przestrzeni w sposób metodyczny i zorganizowany. Zawsze gwarantuje sukces, ale jest bardzo czasochłonne.
3. Szukanie analityczne — wymaga rozwiązania równania matematycznego rządzącego upadkiem szkła kontaktowego z uwzględnieniem oporu powietrza, siły wiatru, ciążenia. Również gwarantuje sukces, ale jest niepraktyczne.
4. Szukanie leniwe — polega na znalezieniu najbliższego optyka i zakupie nowego szkła.
5. Szukanie heurystyczne — określamy przybliżony kierunek upadku i domyślamy się, na jaką odległość może upaść szkło, a następnie przeszukujemy wybrany obszar. Jest to zachowanie najbardziej naturalne i najczęściej nieświadomie wybieramy właśnie ten sposób postępowania.

W przykładzie zamieszczonym wyżej wspomniano o poszukiwaniu ślepym i heurystycznym. O szukaniu ślepym mówimy wtedy, gdy nie wykorzystujemy informacji o dziedzinie rozwiązywanego problemu. W poszukiwaniu heurystycznym korzystamy z dodatkowych informacji o przestrzeni stanów, a ponadto potrafimy ocenić postępy poprawiające efektywność działania. Proces przeszukiwania heurystycznego najlepiej przedstawić w postaci drzewa bądź grafu. W literaturze rozważa się różne strategie przeszukiwania grafu i wyznaczania rozwiązania heurystycznego.

Wracając do szachów, należy stwierdzić, że już po kilku ruchach istnieje tak niewyobrażalna liczba kombinacji, że trudno jest je wszystkie przeanalizować nawet najlepszemu obecnie komputerowi. Stosuje się więc w obecnych programach komputerowych techniki sztucznej inteligencji, a w szczególności specjalnie dobrane i opracowane metody heurystyczne. Dzięki temu komputery szachowe są w stanie nawiązać równorzędną walkę z najlepszymi szachistami świata. Przypomnijmy, że w roku 1996 Garri Kasparow wygrał 4 : 2 mecz z pierwszym modelem komputera Deep Blue. Natomiast w roku następnym przegrał 2,5 : 3,5 z drugim modelem tego komputera o nazwie Deep Blue II. Deep Blue II to supermaszyna szachowa wyprodukowana przez firmę IBM o 32 węzłach, a każdy z nich posiadał kartę z ośmioma wyspecjalizowanymi procesorami szachowymi. Aktualne posunięcia analizowało więc równocześnie 256 procesorów. Taka moc obliczeniowa pozwalała na analizę 200 milionów pozycji na szachownicy w ciągu sekundy. Dodatkowo komputer ten posiadał bazę ze wszystkimi otwarciami z ostatnich 100 lat i bazę z ponad miliardem możliwych końcówek gry. By wygrać z człowiekiem użyto więc niesamowitej mocy obliczeniowej. Po raz kolejny Kasparow spotkał się z maszyną w 2003 roku. Jego przeciwnikiem był tym razem program komputerowy Deep Junior 7. Napisali go dwaj izraelscy programiści — Amir Ban i Shay Bushinsky. Program działał na komputerze z 8 procesorami, znacznie wolniejszym od Deep Blue. Jego wyróżnikiem była większa wiedza na temat szachów. Turniej, który przebiegł od 26 stycznia do 7 lutego w Nowym Jorku zakończył się remisem 3 : 3. Deep Junior 7 analizował od 3 do 7 mln pozycji na szachownicy w ciągu sekundy, Garri Kasparow maksymalnie tylko 3. Wystarczy tylko to porównanie, żeby się przekonać, że daleko komputerom do ludzkiego sposobu myślenia. Jednak człowiekowi nie sprzyja fakt, że szybko się męczy i dodatkowo kierują nim emocje, które też wpływają na wynik gry.

2.7. Kognitywistyka

Kognitywistyka jako nauka istnieje od kilkudziesięciu lat. W roku 1976 zaczęto wydawać kwartalnie pismo *Cognitive Science*, w którym umieszczano wyniki badań naukowych z tej dziedziny, a w 1979 roku powstało towarzystwo naukowe *Cognitive Science Society*, z siedzibą na Uniwersytecie w Michigan. Od tego roku organizowane są też konferencje naukowe, na które zjeżdżają się naukowcy z całego świata. Oprócz nazwy kognitywistyka można spotkać również inne, jak np. nauki kognitywne lub nauki o poznaniu. Kognitywistyka to dziedzina nauki, która próbuje zrozumieć naturę umysłu i zajmuje się zjawiskami dotyczącymi umysłu. Istotną sprawą w naukach kognitywnych jest analiza naszego sposobu postrzegania świata i próba zrozumienia tego, co dzieje się w naszym umyśle, gdy wykonujemy elementarne czynności umysłowe. Wykorzystuje się w tym celu badania nad funkcjonowaniem mózgu oraz modele jego działania. Korzysta się z osiągnięć naukowych neurobiologii i psychologii. Kognitywistyka ma bowiem charakter interdyscyplinarny, na jej potrzeby wykorzystuje się metody i badania również z innych nauk, takich jak antropologia, psychofizyka, sztuczne życie, logika, lingwistyka, neurofizjologia, filozofia, sztuczna inteligencja, i jeszcze z wielu innych gałęzi nauki. Należy stwierdzić, że interdyscyplinarność jest absolutnie niezbędna, aby umożliwić roz-

wój kognitywistyki. Nauka ta zajmuje się niezwykle trudnym problemem badawczym, jakim jest opis funkcjonowania umysłu. Jest oczywiste, że teorie i metody wypracowane w ramach tylko jednej dyscypliny nie doprowadzą do rozwiązania tego problemu. Dlatego efektywne rezultaty mogą uzyskać tylko duże zespoły badawcze, składające się z reprezentantów wyżej wymienionych dyscyplin. Warto dodać, że kognitywistyka ma cały szereg zastosowań praktycznych. Przykładowo, takie dziedziny, jak neurobiologia, psychologia i lingwistyka, wymagają współpracy odpowiednich specjalistów, aby opracować metody leczenia zaburzenia mowy po wylewie krwi do mózgu. Innym obszarem zastosowań są modele kognitywne wykorzystywane do tworzenia interfejsów programów komputerowych. Jedna z koncepcji przewiduje możliwość utworzenia na pulpicie komputera obrazu skojarzeń, jakie mamy w umyśle.

Dużym wyzwaniem nauk kognitywnych jest stworzenie adekwatnych modeli mózgu. Obecne modele w postaci sztucznych sieci neuronowych są niewystarczające i niewiele mają wspólnego z ich rzeczywistym odpowiednikiem. Ponadto badacze zajmujący się kognitywistyką zapewne jeszcze długo będą drążyć problem tzw. słabej i silnej hipotezy w sztucznej inteligencji, o czym wspomnialiśmy w podrozdziale 2.4. W Polsce od 2001 roku działa Polskie Towarzystwo Kognitywistyczne, którego celem jest między innymi promocja zastosowań kognitywistyki oraz wspieranie badań w tej dziedzinie.

2.8. Inteligencja mrówek

Mrówki są owadami, które swoje przeżycie uzależniają przede wszystkim od wspólnego działania. Wielokrotnie obserwowaliśmy mrowisko i dziesiątki mrówek wędrujących chaotycznie w poszukiwaniu pokarmu. Gdy któryś z nich udało się znaleźć jego źródło, wówczas po jakimś czasie jej śladem podążały inne. Naukowców zaciekała fakt, w jaki sposób mrówki znajdują drogę od mrowiska do pożywienia. Okazuje się, że mrówki zazwyczaj wybierają drogę najkrótszą z możliwych. Rozdzielono mrowisko od źródła pożywienia, pozostawiając jako przejście dwa patyczki — dłuższy i krótszy jako jedyną drogę. Po kilku minutach okazało się, że mrówki zaczęły wędrować do pokarmu krótszą drogą. Drugi raz rozpoczęto eksperiment, pozostawiając mrówkom tylko dłuższy patyczek. Oczywiście zaraz znalazły one drogę, ale gdy dostawiono krótszy patyk, ciągle jednak podążały swoją starą drogą. Przyglądając się bliżej ich zachowaniu okazało się, że mrówka podczas swojego marszu zostawia po sobie ślad w postaci substancji zwanej feromonem, tworząc w ten sposób ścieżkę zapachową. Jej towarzyszka, gdy wyczuje taką ścieżkę, podąża jej śladem, oczywiście również pozostawiając po sobie ślad. Wybór drogi kolejnej mrówki uzależniony jest od koncentracji feromonu w danym miejscu. Kierują się one więc tam, gdzie wcześniej przeszło najwięcej współtowarzyszy. Co się jednak stanie, gdy źródło pokarmu się wyczerpie? Okazuje się, że i na to znajdzie się sposób. Feromon po jakimś czasie wyparowuje, tracąc swoją intensywność. Ścieżka mało uczęszczana po pewnym czasie po prostu zniknie. Jeszcze jedno zachowanie mrówek zwróciło szczególną uwagę badaczy. Mrówki usuwają ciała swych martwych towarzyszek, układając je w stosy. Okazuje się, że wystarczy małe zgrupowanie trupów, by w tym samym miejscu pojawiło się owo cmentarzysko. Mrówki i tu kierują się prostą zasadą — przenoszą mar-

two ciała tam, gdzie już leżą inne. To grupowanie mrówek można również wykorzystać praktycznie na przykład w bankowości. Decyzja o udzieleniu komuś kredytu polega na przejrzeniu danych klienta i określeniu, czy jest on wiarygodny, czy nie. Pod uwagę są tu brane takie czynniki, jak wiek, praca, stan cywilny, korzystanie z innych usług banku itp. Analogicznie do zachowania mrówek można tworzyć grupy osób o podobnych cechach. Okazuje się bowiem, że nieuczciwi klienci charakteryzują się zazwyczaj podobnymi cechami. Sprawdzenie klienta polegać będzie zatem na dopasowaniu jego danych do odpowiedniej grupy i sprawdzeniu, czy klienci tam sklasyfikowani byli wiarygodni. Istnieją podobne systemy działające na tej zasadzie, ale przewagą opisanej wyżej metody jest to, że grupy nie są odgórnie narzucone — one tworzą się same.

Na podstawie tych obserwacji opracowany został pewien typ algorytmów, zwany *algorytmami mrówkowymi*. Praca „sztucznych mrówek” stosowanych w tych algorytmach różni się trochę od ich żywych odpowiedników, a mianowicie:

- a) żyją one w sztucznym dyskretnym świecie, poruszają się wobec tego po zupełnie innym terenie, na przykład między wierzchołkami grafu;
- b) ich ślad feromonowy zanika szybciej niż w rzeczywistości;
- c) ilość feromonu wydzielanego przez sztuczną mrówkę uzależniona jest od jakości rozwiązania otrzymanego przez nią;
- d) w większości przypadków ślad feromonowy aktualizowany jest dopiero po wygenerowaniu rozwiązania.

Algorytmy te służą do rozwiązywania trudnych problemów kombinatorycznej optymalizacji, takich jak na przykład problem komiwojażera (ang. TSP — *Traveling Salesman Problem*). Problem komiwojażera polega na tym, że ma on odwiedzić daną liczbę miast jak najkrótszą drogą. Miasta te są różnorodnie odległe od siebie, żadnego z nich nie można pominąć i nie można dwukrotnie znaleźć się w tym samym mieście. Zadanie wydaje się łatwe do rozwiązywania z wykorzystaniem algorytmu sprawdzającego wszystkie warianty. Jednak już przy liczbie kilkunastu miast liczba możliwych dróg rozwija się do rzędu miliardów. Doskonale jednak można wykorzystać w tym przypadku algorytmy mrówkowe, korzystając z pracy „sztucznych mrówek”.

Algorytmy mrówkowe znajdują także zastosowania do rozwiązywania dyskretnych problemów optymalizacyjnych, na przykład do wyznaczania tras pojazdów, sortowania sekwencyjnego czy wyznaczania tras w sieci komputerowej. Praktyczne zastosowania znalazły w sieciach telekomunikacyjnych firm France Telecom i British Telecommunications. Centrale telefoniczne czasami połączone są ze sobą łączami o malej przepustowości. Gdy wzrasta obciążenie sieci, na przykład w czasie konkursu typu audio-tele, łącza te zatykają się. Rozwiązaniem są wirtualni agenci, których praca opiera się na zachowaniu mrówek, dzięki czemu kolejne centrale mogą zwiększać przepustowość poprzez omijanie przeciążonych odcinków sieci.

2.9. Sztuczne życie

Sztuczne życie to młoda dziedzina nauki. Początek wzięła w roku 1987 na konferencji w Santa Fe w Nowym Meksyku (USA), gdzie pojawił się po raz pierwszy termin *Artificial*

Life. Christopher Langton, organizator wspomnianej konferencji, zdefiniował sztuczne życie następująco: „Sztuczne życie jest dziedziną nauki poświęconą zrozumieniu życia poprzez próby wydobycia podstawowych zasad dynamiki, mających wpływ na zjawiska biologiczne. Zjawiska te odtwarzają się za pośrednictwem mediów — na przykład w komputerach — aby móc w pełni wykorzystać nowe metody eksperymentowania”. Jest to dziedzina nauki, która korzysta między innymi z dorobku biologii, chemii, fizyki, psychologii, robotyki oraz nauk komputerowych. Zajmuje się symulacją życia takiego, jakie znamy, ale również są prace, które badają zachowania organizmów zbudowanych na zupełnie innej zasadzie niż istoty ziemskie. Jednak podstawą tej dziedziny nauki jest definicja życia. Niestety w tej kwestii naukowcy nie są zgodni. Główną przyczyną trudności w zdefiniowaniu tego pojęcia jest zapewne fakt, że mamy do czynienia tylko z formami życia spotykanymi na Ziemi.

Najwcześniejszym, prostym i znanym przykładem sztucznego życia jest gra o angielskiej nazwie *Game of Life*. Twórcą gry był w 1968 roku matematyk John Conway, który oparł jej działanie na automatach komórkowych. Środowiskiem w tym przypadku jest dwuwymiarowa tablica komórek, której stan może być określony jako zajęty — reprezentujący żywą komórkę, bądź pusty — brak żywej komórki. Reguły są bardzo proste. Komórka umiera z samotności bądź przeludnienia, a nowa pojawia się wówczas, gdy ma dokładnie trzech sąsiadów. W trakcie symulacji podczas wizualizacji można zaobserwować szybkie, dynamiczne rozrosty komórek, tworzących wspaniałe wzory, a zaraz potem na przykład upadki całych kolonii. Początkowe, nawet niewiele różniące się ustawienia komórek doprowadzają podczas symulacji do bardzo złożonych i ciekawych kształtów.

Innym, ciekawym przykładem sztucznego życia są *biomorfy*. Twórcą jest brytyjski zoolog Richard Dawkins. Powstały one w celu badania ewolucji form. Biomorfy to zapisane w genotypach graficzne kształty, które wyglądem przypominają organizmy żywe. Dawkins zastosował proste operacje genetyczne do uzyskania w kolejnych pokoleniach nowych kształtów. Rozpoczynając ewolucję od prostych figur przypominających drzewa, można uzyskać kształty insektów i owadów. Symulację wzrostu organizmów prowadzono za pomocą formalnego opisu rozwoju tzw. *L-systemów*, zaproponowanych w 1968 roku przez Aristida Lindenmayera. Później stosowano je do opisu i modelowania wzrostu roślin. Efekty były podobne do działania fraktali.

System Tierra to wirtualny świat stworzony przez biologa Toma Raya. Środowiskiem jest tu wirtualny komputer, w którym żyją programy-osobniki. Programy te są pisane w specjalnym, prostym języku przypominającym asembler. Ewolucja osobników przebiega z wykorzystaniem mutacji, czyli losowej zamiany jednej instrukcji, bądź rekombinacji polegającej na podmianie fragmentu kodu. Osobniki rywalizują ze sobą o zasoby, czyli pamięć wirtualnej maszyny, oraz czas wykorzystania procesora. Starają się więc zająć jak najwięcej miejsca oraz wykonywać jak najwięcej swoich instrukcji w porównaniu z innymi osobnikami.

System Framstick jest obecnie jednym z najbardziej zaawansowanych projektów sztucznego życia. Jest on prowadzony od 1997 roku przez Polaków: Macieja Komosińskiego i Szymona Ulatowskiego. Symulacje przeprowadzane są w wirtualnym, trójwymiarowym świecie, gdzie występuje środowisko lądowe i wodne. Organizmy (framsticki) zbudowane są z patyczków, które dodatkowo mogą spełniać funkcje receptorów

(posiadających zmysł dotyku, równowagi lub węchu) lub funkcję narządu ruchu (za pomocą odpowiednich mięśni). Narządy ruchu są sterowane za pomocą układu nerwowego opartego na sieci neuronowej. Framsticci konkurują ze sobą o byt w środowisku poprzez walkę ze sobą i poszukiwanie pożywienia. Każdy z osobników opisany jest genotypem, który jest specjalnym kodem opisującym jego budowę. Dostępne są trzy rodzaje zapisu chromosomu: pierwszy — najprostszy i bezpośrednio opisujący budowę framsticca, drugi ma formę zapisu rekurencyjnego, natomiast trzeci polega na zapisie informacji o konkretnej komórce. Ocena przystosowania przebiega w samym środowisku i może przybrać różne formy. Można oceniać osobnika w zależności od jego poruszania się, wielkości czy też wytrzymałości. Symulacja może również przybierać różne formy, na przykład poszukiwanie najwyższego osobnika. Następne pokolenia osobników powstają poprzez ewolucję, na którą składają się selekcja, mutacja i krzyżowanie genów. Ewolucja dotyczy zarówno zewnętrznej formy organizmu, jak i układu nerwowego.

2.10. Boty

Bot to automat, narzędzie softwarowe, program, służący najczęściej do przeszukiwania i pozyskiwania danych. Inteligentne boty dodatkowo mogą podejmować decyzje na bazie zdobytej wcześniej wiedzy. Obecnie możemy rozróżnić kilka rodzajów botów, które ze względu na umiejętności zostały podzielone na:

- 1) chatterboty — to automaty do pogawędek. Imitują rozmowę w języku naturalnym, pozyskując informacje od rozmówcy;
- 2) searchboty — zajmują się automatyczną obsługą baz danych, służą do przeszukiwania, indeksowania i gromadzenia danych;
- 3) shoppingboty — to automaty, które pomagają przy robieniu zakupów przez Internet. Przeglądają witryny w poszukiwaniu danych produktów, tworząc raport z kształtowania się cen;
- 4) databoty — automaty do przeszukiwania danych, rozwiązywania problemów, ich budowa jest oparta na sieciach neuronowych;
- 5) updateboty — służą do uaktualnienia danych posiadanych przez użytkownika. Informują o zmianach w zasobach sieciowych;
- 6) infoboty — programy automatycznie udzielające odpowiedzi za pomocą poczty elektronicznej. Używane są do obsługi zamówień, pomocy technicznej, informacji marketingowych.

Wśród botów największą popularnością i zainteresowaniem, szczególnie jako narzędzie do badania oczekiwania klienta w marketingu, cieszą się chatterboty. Najczęściej umieszczane są na stronach internetowych, służą do promocji produktów oraz pomagają w nawigacji. Dla firm ich używających są źródłem wiedzy o klientach, w trakcie rozmowy bowiem można pozyskać dużo informacji. Należy uważać, z kim się rozmawia, korzystając z Internetu, może to chatterbot? Użytkownikom Internetu trudno jest zidentyfikować rozmówcę, tym bardziej, że nie spodziewają się konwersacji z modelem. Z ogromnym zdziwieniem dowiadują się, że po drugiej stronie był tylko program komputerowy.

Pierwsze chatterboty pojawiły się ok. 1966 roku jako próby realizacji projektu CMC (ang. *Computer Mediated Communications*), którego celem było nawiązanie łączności człowiek–komputer. W 1968 roku powstała Eliza — program składający się jedynie z 240 linijek kodu, symulujący rozmowę z psychoterapeutą. Program był w stanie zrozumieć strukturę pytania, analizując słowa kluczowe, i na ich podstawie formułować pytania.

Od wielu lat programiści specjalizujący się w pisaniu intelligentnych programów do rozmów startują w specjalnym konkursie. Kilka razy z rzędu najlepszym programistą był Richard Wallace, który stworzył bot o nazwie ALICE (ang. *Artifical Linguistic Internet Computer Entity*). Jest to jeden z najlepszych obecnie botów (www.alicebot.org). Swoje zalety zawdzięcza użytkownikom, z którymi rozmawiał, wzbogacając posiadaną bazę wiedzy. Warto wspomnieć, że nawet tak dobry program nie przeszedł testu Turinga w ramach wspomnianego konkursu. Różne wersje programu ALICE mają komercyjne zastosowania, np. do promocji nowych produktów.

2.11. Perspektywy rozwoju sztucznej inteligencji

Naukowcy od lat spierają się na temat perspektyw rozwoju sztucznej inteligencji. Przytoczymy teraz wypowiedzi znanych specjalistów w tej dziedzinie.

Roger Penrose (profesor matematyki na Uniwersytecie w Oksfordzie) w książce *Nowy umysł cesarza* wyraża przekonanie, że procesy umysłowe człowieka różnią się w sposób fundamentalny od działań komputera. Żadna maszyna pracująca na zasadzie obliczeń nie będzie mogła myśleć i rozumieć tak jak my. Procesy w naszym mózgu są bowiem „nieobliczeniowe”. Ponadto Roger Penrose uważa, że umysł ludzki jest całkowicie wyjaśnialny w kategoriach świata fizycznego, a jedynie istniejące teorie fizyczne nie są wystarczająco kompletne, aby mogły wytlumaczyć, jak przebiegają nasze procesy myślowe. Dr Ray Kurzweil, zajmujący się między innymi komercyjnym wykorzystaniem sztucznej inteligencji, twierdzi, że zanik różnic między maszyną a człowiekiem jest jedynie kwestią czasu, gdyż ludzki mózg w ostatnich wiekach nie rozwinął się prawie wcale. Tymczasem w ostatnim dwudziestoleciu jego elektroniczne odpowiedniki rozwijają się w nieprawdopodobnym tempie i tendencja ta bez wątpienia utrzyma się w następnych latach. Hans Moravec (dyrektor laboratorium mobilnych robotów na Carnegie Mellon University) uważa, że „człowiek jest skomplikowaną maszyną i ... niczym więcej”. Zasłynął śmiały声明, „w przyszłości ludzki system nerwowy da się zastąpić bardziej złożonym, sztucznym odpowiednikiem”. Twierdzi on, że wcześniej czy później kula ziemska będzie zamieszana przez wytwory naszej techniki, doskonalsze i lepiej radzące sobie z trudnościami życia niż wrażliwy i podatny na zranienie przedstawiciel *homo sapiens*. Kevin Warwick (profesor cybernetyki na Uniwersytecie w Reading) twierdzi, że po włączeniu pierwszej potężnej maszyny o inteligencji podobnej do inteligencji człowieka, najprawdopodobniej nie będziemy mogli jej już wyłączyć. Uruchomimy bombę zegarową, cykającą nad ludzkim gatunkiem, i nie zdołamy jej rozbroić. Nie będzie żadnego sposobu powstrzymania marszu maszyn.

Jak widzimy, badania nad sztuczną inteligencją wzbudzają fascynacje badaczy, ale również duże kontrowersje. Podsumowując poglądy i doświadczenia różnych badaczy

oraz eliminując skrajne poglądy, możemy wyrazić następującą opinię:

1) Żadna z maszyn dotychczas stworzonych nie potrafiła wyjść poza zestaw zaprogramowanych przez człowieka zasad.

2) Sztuczne systemy inteligentne nie będą dokładnie symulowały działania ludzkiego mózgu. Wiąże się to z ograniczeniami sprzętowymi oraz z dużą złożonością mózgu.

3) Maszyny mogą przejść test Turinga w wąskim zakresie tematycznym (sport, szachy, system doradczy w ekonomii lub medycynie).

4) W przyszłości może nam się wydawać, że maszyny przejawiają oznaki świadomości. Jednak maszyny nie będą świadome w sensie filozoficznym.

5) W perspektywie kilkudziesięciu lat inteligentne maszyny będą naszymi partnerami w pracy i w domu.

6) Komputery będą projektowały następne generacje komputerów oraz robotów i odegrają znaczącą (dominującą?) rolę w rozwoju inteligencji mieszkańców Ziemi.

2.12. Uwagi

W tym rozdziale omówiliśmy, w sposób przystępny dla czytelnika, wybrane zagadnienia sztucznej inteligencji. Szczegółowe informacje podane są w licznych pozycjach literaturowych na ten temat. Najważniejsze kierunki badań w zakresie inteligencji człowieka omówiono w monografii [143]. Wątki historyczne oraz filozoficzne w dziedzinie sztucznej inteligencji zawierają monografie [105, 168]. Mecze szachowe między maszyną i człowiekiem omówiono w artykułach [81, 124] i [229]. Zagadnienia systemów ekspertowych przedstawiają monografie [5, 25, 89, 141, 146, 170, 179, 268]. Problematykę budowy robotów oraz ich zastosowań poruszają autorzy książki [61] oraz [135]. Zagadnienia przetwarzania mowy i języka naturalnego omówiono w pracach [45, 154, 228]. Ideę przeszukiwania heurystycznego i stosowane algorytmy wyjaśniono w pracach [62, 70]. Historię i rozwój kognitywistyki przedstawiono w artykule [46]. Zachowanie mrówek oraz algorytmy mrówkowe wyjaśniono w pracy [14]. Różne modele sztucznego życia omówiono w artykułach [114, 169]. W pracy [75] przedstawiono zastosowania sztucznej inteligencji w elektroenergetyce, w pracy [170] w diagnostyce technicznej, w pracy [268] w ekonomii i zarządzaniu. Godne polecenia są klasyczne w literaturze światowej monografie na temat sztucznej inteligencji [129, 147, 184]. W monografii [245] autorzy przedstawili niezwykle interesujące podejście do zagadnienia rozpoznawania obrazów metodami sztucznej inteligencji, a w szczególności wprowadzili pojęcie tzw. „automatycznego rozumienia obrazów”. W tym rozdziale poruszyliśmy tylko niektóre wątki sztucznej inteligencji. Czytelnika zainteresowanego na przykład wieloagentowymi systemami inteligentnymi odsyłamy do monografii [52, 109, 127, 255]. Z kolei monografia [27] jest obszernym opracowaniem na temat systemów uczących się i prezentuje klasyczne metody sztucznej inteligencji, jak np. zagadnienia indukcji drzew decyzyjnych i indukcji reguł, metody probabilistyczne oraz indukcyjne programowanie logiczne.

Metody reprezentacji wiedzy z wykorzystaniem zbiorów przybliżonych

3.1. Wprowadzenie

W otaczającym nas fizycznym świecie nie znajdziemy dwóch obiektów (przedmiotów) identycznych. Porównując dowolne dwa przedmioty, choćby bardzo podobne, zawsze możemy znaleźć między nimi różnice, zwłaszcza jeśli uwzględnimy dostatecznie dużo ich cech (atrybutów), z dostatecznie dużą dokładnością. Oczywiście nie zawsze konieczny jest tak drobiazgowy opis świata. Jeśli zmniejszymy dokładność opisu, to może się zdarzyć, że kilka lub kilkaset obiektów, uprzednio rozróżnialnych, stanie się obiektami nieroróżnialnymi. Na przykład, wszystkie miasta w Polsce mogą być rozróżnialne ze względu na dokładną liczbę mieszkańców. Jeśli interesują nas miasta o liczbie mieszkańców znajdującej się w konkretnym przedziale, np. od 100 do 300 tys., to część miast będzie nieroróżnialna pod względem cechy (atrybutu) „liczba mieszkańców”. Ponadto, w opisie danego obiektu ograniczamy się do określonej liczby cech, adekwatnej do danego zastosowania. Nierzadko zależy nam na zredukowaniu tej liczby do niezbędnego minimum. Tymi oto problemami zajmuje się teoria zbiorów przybliżonych.

W celu ułatwienia dalszych rozważań wprowadźmy kilka pojęć i oznaczeń. Jako pierwszą określmy przestrzeń rozważań U . Jest to zbiór wszystkich obiektów, które są w kręgu naszych zainteresowań. Pojedynczy j -y element tej przestrzeni oznaczmy jako x_j . Każdy obiekt przestrzeni U może być scharakteryzowany za pomocą pewnych cech. Jeśli jest to obiekt fizyczny, to ma zapewne tych cech nieskończonie wiele, my ograniczmy się jednak do pewnego ich podzbioru. Oznaczmy zbiór interesujących nas cech obiektów przestrzeni U symbolem Q . Poszczególne cechy oznaczmy symbolami q z odpowiednimi indeksami, np. q_i . Tym, co odróżnia jeden obiekt od drugiego, a inne obiekty czyni podobnymi, są wartości ich cech. Oznaczmy przez V_q zbiór wartości, jakie może przyjmować cecha q . Wartość cechy q obiektu x oznaczmy jako v_q^x . Wektor wszystkich cech obiektu x możemy przedstawić jako $\mathbf{v}^x = [v_{q_1}^x, v_{q_2}^x, \dots, v_{q_n}^x]$.

Informacje z dziedziny zbiorów przybliżonych zostaną przedstawione poniżej w postaci szeregu definicji ilustrowanych przykładami. Tabela 3.1 ułatwi czytelnikowi poruszenie się między nimi.

Tabela 3.1. Spis definicji przykładów

Definicja	Przykłady
3.1. System informacyjny	3.1, 3.2, 3.3
3.2. Tablica decyzyjna	3.4
3.3. Relacja nieroróżnialności	
3.4. Klasa abstrakcji	3.5, 3.6, 3.7
3.5. Dolna aproksymacja zbioru	3.8, 3.9, 3.10
3.6. Górska aproksymacja zbioru	3.11, 3.12, 3.13
3.7. Pozytywny obszar zbioru	3.8, 3.9, 3.10
3.8. Brzegowy obszar zbioru	3.14, 3.15, 3.16
3.9. Negatywny obszar zbioru	3.17, 3.18, 3.19
3.10. Zbiór \tilde{P} -dokładny	3.20, 3.21, 3.22
3.11. Zbiór \tilde{P} -definiowalny	3.20, 3.21, 3.22
3.12. \tilde{P} -dokładność zbioru	3.23, 3.24, 3.25
3.13. Dolna aproksymacja rodziny zbiorów	3.26
3.14. Górska aproksymacja rodziny zbiorów	3.27
3.15. Pozytywny obszar rodziny zbiorów	3.28
3.16. Brzegowy obszar rodziny zbiorów	3.29
3.17. Negatywny obszar rodziny zbiorów	3.30
3.18. Jakość aproksymacji rodziny zbiorów	3.31
3.19. Dokładność aproksymacji rodziny zbiorów	3.32
3.20. Stopień zależności atrybutów	3.33
3.21. Reguły deterministyczne	3.33
3.22. Zbiór atrybutów niezależnych	3.34
3.23. Zbiór atrybutów względnie niezależnych	3.35
3.24. Redukt	3.36
3.25. Redukt względny	3.36
3.26. Atrybut nieusuwalny	3.37
3.27. Rdzeń	3.38
3.28. Znormalizowany współczynnik istotności atrybutów	3.39
3.29. Błąd reduktu przybliżonego	3.40

3.2. Pojęcia podstawowe

Jednym ze sposobów przedstawienia informacji o obiektach charakteryzowanych przez ten sam zbiór cech jest system informacyjny.

DEFINICJA 3.1

Systemem informacyjnym nazywamy uporządkowaną czwórkę $SI = (U, Q, V, f)$, gdzie U jest zbiorem obiektów, Q jest zbiorem cech (atributów), $V = \bigcup_{q \in Q} V_q$ jest zbiorem wszystkich możliwych wartości cech, natomiast $f : U \times Q \rightarrow V$ nazywamy funkcją informacyjną. Możemy zapisać, że $v_q^x = f(x, q)$, oczywiście $f(x, q) \in V_q$. Za równorzędnny będziemy uważać zapis $v_q^x = f_x(q)$, traktujący funkcję informacyjną jako rodzinę funkcji. Wówczas $f_x : Q \rightarrow V$.

Przykład 3.1

Rozważmy komis samochodowy. W chwili obecnej znajduje się w nim 10 pojazdów. Przestrzeń rozważań U składa się więc z 10 obiektów, co możemy zapisać

$$U = \{x_1, x_2, \dots, x_{10}\}. \quad (3.1)$$

Właściciel komisu notuje w swoich dokumentach cztery cechy każdego samochodu, o które zwykle pytają klienci w rozmowach telefonicznych. Są to: liczba drzwi, moc silnika, kolor i marka. Zatem zbiór cech możemy zapisać

$$\begin{aligned} Q &= \{q_1, q_2, q_3, q_4\} \\ &= \{\text{liczba drzwi, moc silnika, kolor, marka}\}. \end{aligned} \quad (3.2)$$

Tabela 3.2 przedstawia notatki właściciela komisu, które tworzą przykład systemu informacyjnego.

Tabela 3.2. Przykład systemu informacyjnego

Obiekt (U)	Liczba drzwi (q_1)	Moc silnika (q_2)	Kolor (q_3)	Marka (q_4)
x_1	2	60	niebieski	Opel
x_2	2	100	czarny	Nissan
x_3	2	200	czarny	Ferrari
x_4	2	200	czerwony	Ferrari
x_5	2	200	czerwony	Opel
x_6	3	100	czerwony	Opel
x_7	3	100	czerwony	Opel
x_8	3	200	czarny	Ferrari
x_9	4	100	niebieski	Nissan
x_{10}	4	100	niebieski	Nissan

Na podstawie zawartości tabeli 3.2 możemy określić dziedziny poszczególnych cech:

$$V_{q_1} = \{2, 3, 4\}, \quad (3.3)$$

$$V_{q_2} = \{60, 100, 200\}, \quad (3.4)$$

$$V_{q_3} = \{\text{czarny, niebieski, czerwony}\}, \quad (3.5)$$

$$V_{q_4} = \{\text{Ferrari, Nissan, Opel}\}. \quad (3.6)$$

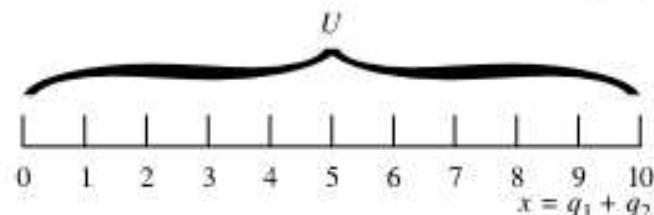
Przykład 3.2

Rozważmy zbiór liczb rzeczywistych w przedziale U (rys. 3.1), gdzie

$$U = [0, 10]. \quad (3.7)$$

Niech każdy element $x \in U$ będzie określony przez dwie cechy tworzące zbiór cech

$$Q = \{q_1, q_2\}, \quad (3.8)$$



Rys. 3.1. Jednowymiarowa przestrzeń rozważań

gdzie q_1 jest częścią całkowitą liczby x , a q_2 jest częścią ułamkową tej liczby. Oczywiście $x = q_1 + q_2$. Funkcje informacyjne możemy zdefiniować następująco:

$$f_x(q_1) = \text{Ent}(x), \quad (3.9)$$

$$f_x(q_2) = x - \text{Ent}(x), \quad (3.10)$$

gdzie funkcja $\text{Ent}(\cdot)$ (fr. *entier*) oznacza część całkowitą argumentu.

Znając definicje funkcji informacyjnych na ogół nietrudno określić dziedziny zmienności poszczególnych cech. W naszym przykładzie będą one następujące:

$$V_{q_1} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \quad (3.11)$$

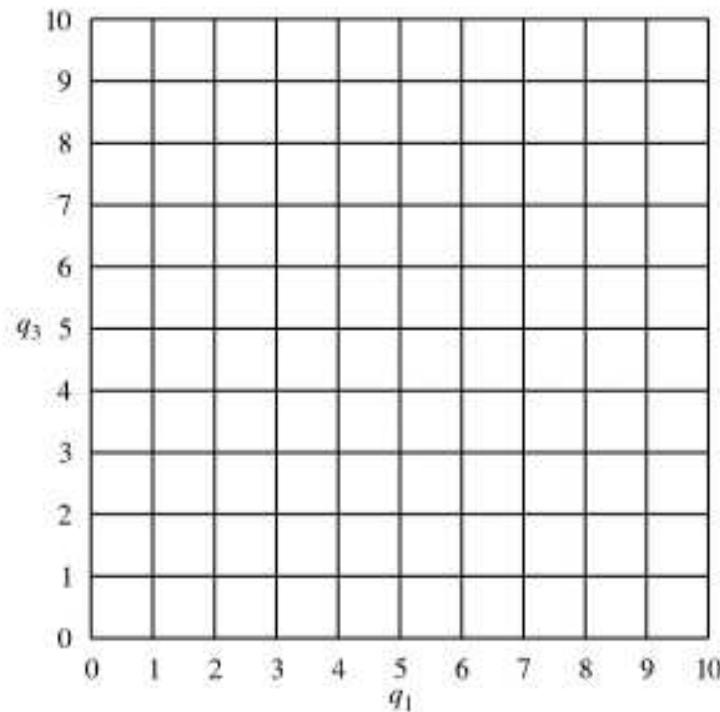
$$V_{q_2} = [0; 1). \quad (3.12)$$

Przykład 3.3

W kolejnym przykładzie rozważmy przestrzeń par

$$U = \{\mathbf{x} = [x_1; x_2] \in [0; 10) \times [0; 10)\}. \quad (3.13)$$

Obiekty należące do tak zdefiniowanej przestrzeni możemy interpretować jako punkty leżące na płaszczyźnie, jak pokazano na rysunku 3.2. Najbardziej naturalnymi cechami



Rys. 3.2. Dwuwymiarowa przestrzeń rozważań

punktów są ich współrzędne x_1 i x_2 . W naszym przykładzie zdefiniujemy je jednak odmiennie. Określmy cztery cechy

$$Q = \{q_1, q_2, q_3, q_4\}, \quad (3.14)$$

gdzie q_1 jest częścią całkowitą pierwszej współrzędnej punktu \mathbf{x} , q_2 jest jej częścią ułamkową, a q_3 i q_4 są, odpowiednio, częścią całkowitą i ułamkową drugiej współrzędnej punktu. Funkcje informacyjne będą więc zdefiniowane następująco:

$$f_{\mathbf{x}}(q_1) = \text{Ent}(x_1), \quad (3.15)$$

$$f_{\mathbf{x}}(q_2) = x_1 - \text{Ent}(x_1), \quad (3.16)$$

$$f_{\mathbf{x}}(q_3) = \text{Ent}(x_2), \quad (3.17)$$

$$f_{\mathbf{x}}(q_4) = x_2 - \text{Ent}(x_2). \quad (3.18)$$

Znając definicje funkcji informacyjnych, na ogół nietrudno określić dziedziny zmienności poszczególnych cech. W tym przykładzie będą one następujące:

$$V_{q_1} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \quad (3.19)$$

$$V_{q_2} = [0; 1], \quad (3.20)$$

$$V_{q_3} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \quad (3.21)$$

$$V_{q_4} = [0; 1]. \quad (3.22)$$

Szczególnym przypadkiem systemu informacyjnego są tablice decyzyjne.

DEFINICJA 3.2

Tablicą decyzyjną nazywamy uporządkowaną piątkę $DT = (U, C, D, V, f)$. Elementy zbioru C nazywamy cechami (atrybutami) warunkowymi, a elementy D — cechami (atrybutami) decyzyjnymi.

Funkcja informacyjna f opisana w definicji 3.1 określa jednoznacznie zbiór reguł zawarty w tablicy decyzyjnej. W zapisie, w postaci rodziny funkcji, funkcja $f_l: C \times D \rightarrow V$ definiuje l -tą regułę decyzyjną tablicy. Różnica między powyższą definicją a definicją 3.1 polega na rozdzieleniu zbioru cech Q na dwa rozłączne podzbiory C i D , dopełniające się do Q . Tablice decyzyjne stanowią alternatywny sposób przedstawiania informacji w stosunku do reguł:

$$R^l : \mathbf{JEŻELI} \, c_1 = v_{c_1}^l \, \mathbf{I} \, c_2 = v_{c_2}^l \, \mathbf{I} \dots \mathbf{I} \, c_{n_c} = v_{c_{n_c}}^l \, \mathbf{TO} \, d_1 = v_{d_1}^l \, \mathbf{I} \, d_2 = v_{d_2}^l \, \mathbf{I} \dots \mathbf{I} \, d_{n_d} = v_{d_{n_d}}^l$$

Przykład 3.4

Założymy, że opierając się na notatkach właściciela komisu z przykładu 3.1, zbudujemy system ekspertowy, który na podstawie informacji o liczbie drzwi, mocy silnika i kolorze samochodu określi jego markę. Dokonaliśmy zatem podziału zbioru Q (zdefiniowanego wzorem (3.2)) na zbiór cech warunkowych

$$\begin{aligned} C &= \{c_1, c_2, c_3\} = \{q_1, q_2, q_3\} \\ &= \{\text{liczba drzwi, moc silnika, kolor}\} \end{aligned} \quad (3.23)$$

oraz jednoelementowy zbiór cech decyzyjnych

$$D = \{d_1\} = \{q_4\} = \{\text{marka}\}. \quad (3.24)$$

Informacje zawarte w systemie informacyjnym przedstawionym w tabeli 3.2 wykorzystamy do budowy tablicy decyzyjnej (tab. 3.3). Opis każdego obiektu przestrzeni U stanowi podstawę stworzenia jednej reguły.

Tabela 3.3. Przykład tablicy decyzyjnej

Reguła (l)	Liczba drzwi (c ₁)	Moc silnika (c ₂)	Kolor (c ₃)	Marka (d ₁)
1	2	60	niebieski	Opel
2	2	100	czarny	Nissan
3	2	200	czarny	Ferrari
4	2	200	czerwony	Ferrari
5	2	200	czerwony	Opel
6	3	100	czerwony	Opel
7	3	100	czerwony	Opel
8	3	200	czarny	Ferrari
9	4	100	niebieski	Nissan
10	4	100	niebieski	Nissan

Treść zawartą w tablicy decyzyjnej (tab. 3.3) można przedstawić również w postaci reguł:

R^1 : **JEŻELI** $c_1 = 2$ **I** $c_2 = 60$ **I** $c_3 = \text{niebieski}$ **TO** $d_1 = \text{Opel}$

R^2 : **JEŻELI** $c_1 = 2$ **I** $c_2 = 100$ **I** $c_3 = \text{czarny}$ **TO** $d_1 = \text{Nissan}$

...

R^{10} : **JEŻELI** $c_1 = 4$ **I** $c_2 = 100$ **I** $c_3 = \text{niebieski}$ **TO** $d_1 = \text{Nissan}$

Podamy teraz dwie definicje niezwykle istotne w teorii zbiorów przybliżonych. Jeśli pewne dwa obiekty $x_a, x_b \in U$ mają takie same wartości wszystkich cech q należących do zbioru $P \subseteq Q$, co możemy zapisać $\forall q \in P, f_{x_a}(q) = f_{x_b}(q)$, to mówimy, że obiekty te są P -nierozróżnialne lub że są ze sobą w relacji P -nierozróżnialności ($x_a \tilde{P} x_b$).

DEFINICJA 3.3

Relację P -nierozróżnialności nazywamy relację \tilde{P} określoną na przestrzeni $U \times U$, spełniającą zależność

$$x_a \tilde{P} x_b \Leftrightarrow \forall q \in P; f_{x_a}(q) = f_{x_b}(q), \quad (3.25)$$

w której $x_a, x_b \in U, P \subseteq Q$.

Łatwo sprawdzić, że relacja \tilde{P} jest zwrotna, symetryczna i przechodnia, a więc jest relacją równoważności. Relacja równoważności dzieli zbiór, w którym jest określona, na rodzinę rozłącznych podzbiorów zwanych klasami abstrakcji tej relacji.

DEFINICJA 3.4

Klasą abstrakcji relacji \tilde{P} w przestrzeni U nazywamy zbiór wszystkich obiektów $x \in U$ będących w relacji \tilde{P} . Dla każdego $x_a \in U$ istnieje dokładnie jeden taki zbiór oznaczony symbolem $[x_a]_{\tilde{P}}$, tzn.

$$[x_a]_{\tilde{P}} = \{x \in U : x_a \tilde{P} x\}. \quad (3.26)$$

Rodzinę wszystkich klas abstrakcji relacji \tilde{P} w przestrzeni U (zwaną ilorazem zbioru U przez relację \tilde{P}) oznaczamy przez P^* lub U/\tilde{P} .

Przykład 3.5

Wyznaczmy klasy abstrakcji relacji C -nierozróżnialności \tilde{C} określonej przez zbiór cech C dany wzorem (3.23) dla systemu informacyjnego zdefiniowanego w przykładzie 3.1:

$$[x_1]_{\tilde{C}} = \{x_1\}, \quad (3.27)$$

$$[x_2]_{\tilde{C}} = \{x_2\}, \quad (3.28)$$

$$[x_3]_{\tilde{C}} = \{x_3\}, \quad (3.29)$$

$$[x_4]_{\tilde{C}} = [x_5]_{\tilde{C}} = \{x_4, x_5\}, \quad (3.30)$$

$$[x_6]_{\tilde{C}} = [x_7]_{\tilde{C}} = \{x_6, x_7\}, \quad (3.31)$$

$$[x_8]_{\tilde{C}} = \{x_8\}, \quad (3.32)$$

$$[x_9]_{\tilde{C}} = [x_{10}]_{\tilde{C}} = \{x_9, x_{10}\}. \quad (3.33)$$

Możemy zatem powiedzieć, że obiekty x_4 i x_5 są C -nierozróżnialne, podobnie jak x_6 i x_7 oraz x_9 i x_{10} . Rodziną wymienionych klas abstrakcji będzie zbiór

$$C^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4, x_5\}, \{x_6, x_7\}, \{x_8\}, \{x_9, x_{10}\}\}. \quad (3.34)$$

Przykład 3.6

W zbiorze cech $Q = \{q_1, q_2\}$ zdefiniowanych w przykładzie 3.2 wszystkie obiekty są rozróżnialne, tzn. istnieje nieskończenie wiele jednoelementowych klas abstrakcji relacji Q -nierozróżnialności i każdy element przestrzeni U tworzy własną klasę. Inaczej będzie, gdy podzielimy zbiór Q na dwa zbiory cech:

$$P_1 = \{q_1\}, \quad (3.35)$$

$$P_2 = \{q_2\}. \quad (3.36)$$

Dla relacji P_1 -nierozróżnialności powstaje 10 klas abstrakcji

$$[0]_{P_1} = [0; 1), \quad (3.37)$$

$$[1]_{P_1} = [1; 2), \quad (3.38)$$

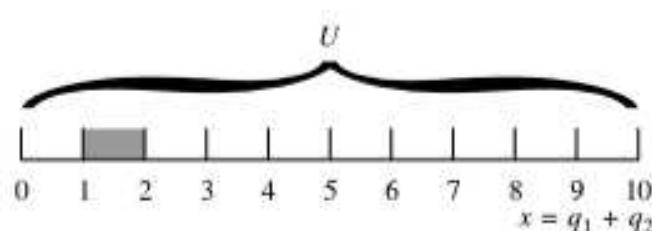
...

$$[9]_{P_1} = [9; 10). \quad (3.39)$$

Ich rodziną jest zbiór

$$P_1^* = \{[0; 1); [1; 2); [2; 3); [3; 4); [4; 5); [5; 6); [6; 7); [7; 8); [8; 9); [9; 10)\}. \quad (3.40)$$

Na rysunku 3.3 pokazano przykładową klasę abstrakcji $[1]_{\tilde{P}_1}$.



Rys. 3.3. Przykład klasy abstrakcji

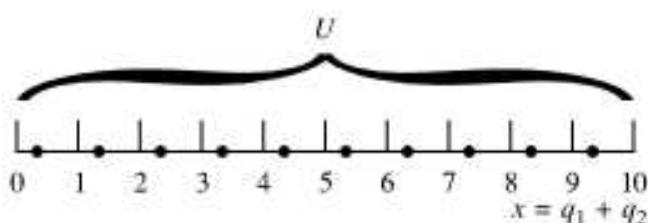
Dla relacji P_2 -nierozróżnialności powstaje nieskończoność wiele dziesięcioelementowych klas abstrakcji. Są to zbiory liczb z przestrzeni U o takiej samej części ułamkowej

$$[x]_{P_2} = \{\hat{x} \in U : \hat{x} - \text{Ent}(\hat{x}) = x - \text{Ent}(x)\}. \quad (3.41)$$

Ich rodziną jest zbiór

$$\begin{aligned} P_2^* &= \{[x]_{P_2} = \{\hat{x} \in U : \hat{x} - \text{Ent}(\hat{x}) = x - \text{Ent}(x)\} : x \in [0; 1)\} \\ &= \{[x]_{P_2} = \{\hat{x} \in U : \hat{x} - \text{Ent}(\hat{x}) = x\} : x \in [0; 1)\}. \end{aligned} \quad (3.42)$$

Na rysunku 3.4 pokazano przykładową klasę abstrakcji $[0,33]_{\tilde{P}_2}$.



Rys. 3.4. Przykład klasy abstrakcji

Przykład 3.7

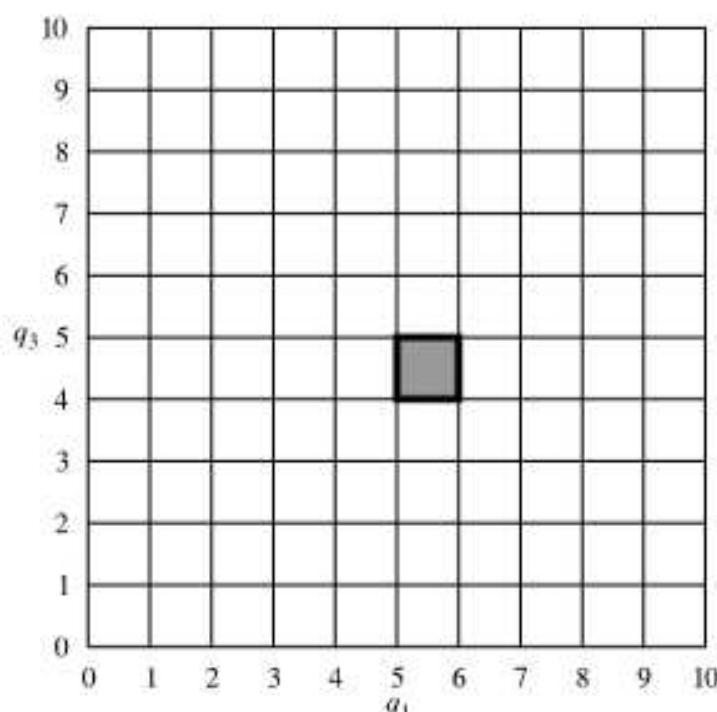
Podobnie jak w przykładzie 3.6, w zbiorze cech Q , określonych wzorem (3.14) w przykładzie 3.3, wszystkie elementy są rozróżnialne, tzn. istnieje nieskończoność wiele jednoelementowych klas abstrakcji relacji Q -nierozróżnialności i każdy element przestrzeni U tworzy własną klasę. Inaczej będzie, gdy rozważymy podzbiór cech $P \subseteq Q$, np.

$$P = \{q_1, q_3\}. \quad (3.43)$$

Dla określonej w ten sposób relacji P -nierozróżnialności w przestrzeni U otrzymujemy 100 klas abstrakcji. Klasę abstrakcji punktu $\mathbf{x} = (x_1, x_2)$ można opisać jako

$$[\mathbf{x}]_{\tilde{P}} = \{\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2) \in U : \text{Ent}(\tilde{x}_1) = \text{Ent}(x_1) \wedge \text{Ent}(\tilde{x}_2) = \text{Ent}(x_2)\}. \quad (3.44)$$

Rysunek 3.5 przedstawia przykładową klasę abstrakcji.



Rys. 3.5. Przykład klasy abstrakcji

Ich rodziną jest zbiór wszystkich kwadratowych pól widocznych na rysunkach 3.2 i 3.5. Możemy ten zbiór opisać następująco:

$$\begin{aligned}
 P^* = & \{[\mathbf{x}]_{\tilde{P}} = \{\tilde{\mathbf{x}} = (\tilde{x}_1; \tilde{x}_2) \in U : \text{Ent}(\tilde{x}_1) = \text{Ent}(x_1) \wedge \text{Ent}(\tilde{x}_2) = \text{Ent}(x_2)\} : \\
 & \mathbf{x} = (x_1; x_2); x_1; x_2 = 0; \dots; 9\} \\
 = & \{[\mathbf{x}]_{\tilde{P}} = \{\tilde{\mathbf{x}} = (\tilde{x}_1; \tilde{x}_2) \in U : \text{Ent}(\tilde{x}_1) = x_1 \wedge \text{Ent}(\tilde{x}_2) = x_2\} : \\
 & \mathbf{x} = (x_1; x_2); x_1; x_2 = 0; \dots; 9\}.
 \end{aligned} \tag{3.45}$$

3.3. Aproksymacja zbioru

W przestrzeni U mogą istnieć pewne zbiory X . O przynależności poszczególnych obiektów $x \in U$ do zbiorów X wnioskujemy na podstawie znajomości wartości ich cech. Zbiór dostępnych cech $P \subseteq Q$ jest zwykle ograniczony i określenie przynależności obiektu do konkretnego zbioru może nie być jednoznaczne. Sytuację tę opisują pojęcia dolnej i górnej aproksymacji zbioru $X \subseteq U$.

DEFINICJA 3.5

\tilde{P} -dolną aproksymacją zbioru $X \subseteq U$ nazywamy zbiór $\underline{\tilde{P}}X$ opisany następująco:

$$\underline{\tilde{P}}X = \{x \in U : [x]_{\tilde{P}} \subseteq X\}. \tag{3.46}$$

Zatem dolną aproksymacją zbioru X jest zbiór tych obiektów $x \in U$, o których na podstawie wartości cech P możemy z całą pewnością powiedzieć, że są elementami zbioru X .

Przykład 3.8

W przestrzeni U , zdefiniowanej równaniem (3.1) w przykładzie 3.1, istnieją trzy zbiory samochodów marek Ferrari, Nissan i Opel (patrz tab. 3.2). Oznaczmy je literami X_F , X_N i X_O :

$$X_F = \{x_3, x_4, x_8\}, \tag{3.47}$$

$$X_N = \{x_2, x_9, x_{10}\}, \tag{3.48}$$

$$X_O = \{x_1, x_5, x_6, x_7\}. \tag{3.49}$$

O przynależności obiektów przestrzeni będziemy wnioskować na podstawie wartości cech ze zbioru C zdefiniowanego zapisem (3.23). Korzystając bezpośrednio z definicji 3.5, określmy \tilde{C} -dolną aproksymację zbiorów X_F , X_N i X_O . Definicja ta mówi, że obiekt $x \in U$ jest elementem dolnej aproksymacji, jeżeli cała klasa abstrakcji, do której on należy, jest podzbiorem zbioru X . Wśród klas abstrakcji wyznaczonych w przykładzie 3.5 tylko klasy $[x_3]_{\tilde{C}}$ i $[x_8]_{\tilde{C}}$ są podzbiorami zbioru X_F , czyli

$$\underline{\tilde{C}}X_F = \{x_3\} \cup \{x_8\} = \{x_3, x_8\}. \tag{3.50}$$

Obiekt x_4 nie należy do $\underline{\tilde{C}}X_F$, mimo iż należy do X_F , gdyż obiekt x_5 o identycznych wartościach cech ze zbioru C , a więc należący do tej samej klasy abstrakcji, nie jest elementem X_F .

Zbiory $[x_2]_{\tilde{C}}$ oraz $[x_9]_{\tilde{C}} = [x_{10}]_{\tilde{C}}$ są podzbiorami zbioru X_N , zatem

$$\tilde{C}X_N = \{x_2\} \cup \{x_9, x_{10}\} = \{x_2, x_9, x_{10}\}. \quad (3.51)$$

Zbiory $[x_1]_{\tilde{C}}$ i $[x_6]_{\tilde{C}} = [x_7]_{\tilde{C}}$ są podzbiorami zbioru X_O , a więc

$$\tilde{C}X_O = \{x_1\} \cup \{x_6, x_7\} = \{x_1, x_6, x_7\}. \quad (3.52)$$

Przykład 3.9

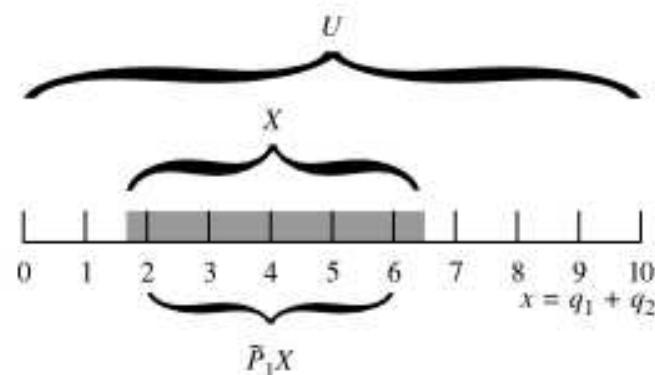
Przyjmijmy, że w przestrzeni U zdefiniowanej w przykładzie 3.2 istnieje zbiór X określony następująco:

$$X = [1,75; 6,50]. \quad (3.53)$$

Określmy \tilde{P}_1 oraz \tilde{P}_2 -dolną aproksymację tego zbioru. Cztery klasy abstrakcji relacji P_1 -nierozróżnialności (przykład 3.6) należą w całości do zbioru X . Zatem P_1 -dolna aproksymacja będzie ich sumą

$$\tilde{P}_1 X = [2]_{P_1} \cup [3]_{P_1} \cup [4]_{P_1} \cup [5]_{P_1} = [2, 6], \quad (3.54)$$

co ilustruje rysunek 3.6.



Rys. 3.6. Dolna aproksymacja zbioru w jednowymiarowej przestrzeni rozważań

Żadna klasa abstrakcji relacji P_2 -nierozróżnialności nie należy w całości do zbioru X , więc jego \tilde{P}_2 -dolna aproksymacja jest zbiorem pustym, tzn.

$$\tilde{P}_2 X = \emptyset. \quad (3.55)$$

Przykład 3.10

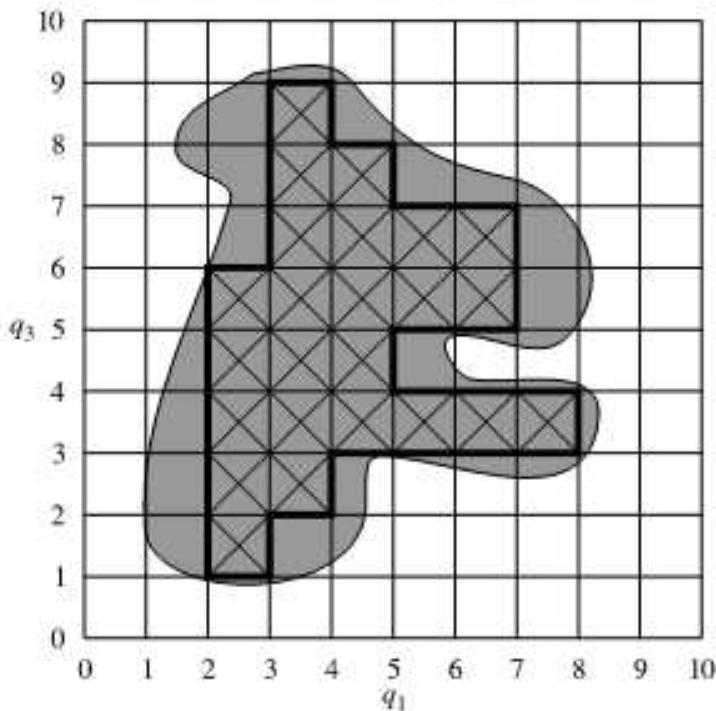
Niech w przestrzeni U określonej zapisem (3.13) będzie zdefiniowany zbiór X , jak pokazano na rysunku 3.7. Na rysunku tym są zakreślone klasy abstrakcji wchodzące w skład \tilde{P} -dolnej aproksymacji zbioru X . Spośród 100 klas abstrakcji określonych wzorem (3.44), dolną aproksymację tworzy 25 klas abstrakcji — kwadratów, które w całości są podzbiorami zbioru X .

DEFINICJA 3.6

\tilde{P} -górną aproksymacją zbioru $X \subseteq U$ nazywamy zbiór $\tilde{P}X$ opisany następująco:

$$\tilde{P}X = \{x \in U : [x]_{\tilde{P}} \cap X \neq \emptyset\}. \quad (3.56)$$

Górną aproksymacją zbioru X jest zbiór tych obiektów $x \in U$, o których na podstawie wartości cech P nie możemy z całą pewnością powiedzieć, że nie są elementami zbioru X .



Rys. 3.7. Dolna aproksymacja zbioru w dwuwymiarowej przestrzeni rozważań

Przykład 3.11

Korzystając bezpośrednio z definicji 3.6, określmy \tilde{C} -górna aproksymację zbiorów X_F , X_N i X_O zdefiniowanych w przykładzie 3.8. Definicja ta mówi, że obiekt $x \in X$ jest elementem górnej aproksymacji, jeżeli klasa abstrakcji, do której on należy, ma niepustą część wspólną ze zbiorem X . Innymi słowy, jeżeli choćby jeden element danej klasy abstrakcji należy do zbioru X , to do górnej aproksymacji zbioru X należy każdy element tej klasy abstrakcji. Wśród klas abstrakcji wyznaczonych w przykładzie 3.5, elementy klas $[x_3]_{\tilde{C}}$, $[x_4]_{\tilde{C}}$ i $[x_8]_{\tilde{C}}$ należą do zbioru X_F , czyli

$$\tilde{C}X_F = \{x_3\} \cup \{x_4, x_5\} \cup \{x_8\} = \{x_3, x_4, x_5, x_8\}. \quad (3.57)$$

Obiekt x_5 należy do \tilde{C} -górnej aproksymacji zbioru X_F , mimo iż nie należy do zbioru X_F , gdyż obiekt x_4 o identycznych wartościach cech ze zbioru C , a więc należący do tej samej klasy abstrakcji, jest elementem zbioru X_F . Do zbioru X_N należą elementy z klas $[x_2]_{\tilde{C}}$ oraz $[x_2]_{\tilde{C}} = [x_{10}]_{\tilde{C}}$, zatem

$$\tilde{C}X_N = \{x_2\} \cup \{x_9, x_{10}\} = \{x_2, x_9, x_{10}\}. \quad (3.58)$$

Do zbioru X_O należą obiekty z klas $[x_1]_{\tilde{C}}$, $[x_4]_{\tilde{C}}$ i $[x_6]_{\tilde{C}}$, skąd

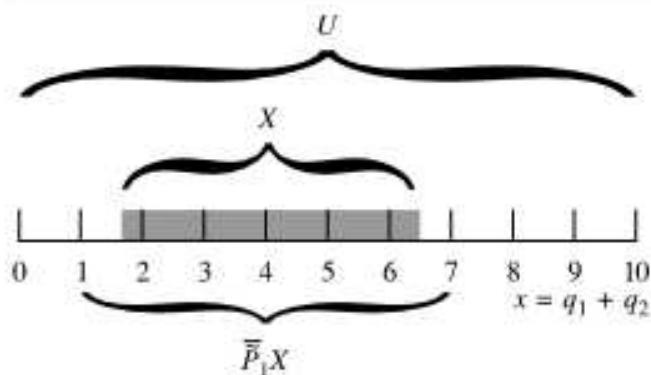
$$\tilde{C}X_O = \{x_1\} \cup \{x_4, x_5\} \cup \{x_6, x_7\} = \{x_1, x_4, x_5, x_6, x_7\}. \quad (3.59)$$

Przykład 3.12

Określmy \tilde{P}_1 - i \tilde{P}_2 -górna aproksymację zbioru X zdefiniowanego w przykładzie 3.9. Obiekty sześciu klas abstrakcji P_1 -nieroróżnialności należą do zbioru X . Zatem \tilde{P}_1 -górna aproksymacja będzie ich sumą

$$\tilde{P}_1X = [1]_{\tilde{P}_1} \cup [2]_{\tilde{P}_1} \cup [3]_{\tilde{P}_1} \cup [4]_{\tilde{P}_1} \cup [5]_{\tilde{P}_1} \cup [6]_{\tilde{P}_1} = [1, 7], \quad (3.60)$$

co obrazuje rysunek 3.8.



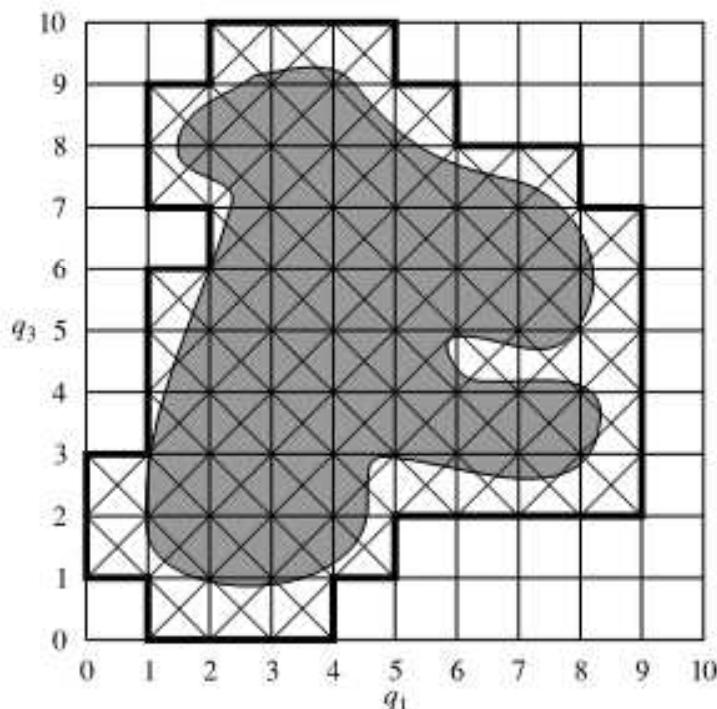
Rys. 3.8. Górnna aproksymacja zbioru w jednowymiarowej przestrzeni rozważan

Ponieważ elementy wszystkich klas abstrakcji relacji P_2 -nieroróżnialności należą do zbioru X , więc jego \tilde{P}_2 -górnna aproksymacja pokrywa się z przestrzenią rozważan U , tzn.

$$\tilde{P}_2 X = U. \quad (3.61)$$

Przykład 3.13

Na rysunku 3.9 zakreślone są klasy abstrakcji wchodzące w skład P -górnnej aproksymacji zbioru X określonego w przestrzeni U zdefiniowanej wzorem (3.13).



Rys. 3.9. Górnna aproksymacja zbioru w dwuwymiarowej przestrzeni rozważan

DEFINICJA 3.7

\tilde{P} -pozytywny obszar zbioru X definiujemy jako

$$\text{Pos}_{\tilde{P}}(X) = \underline{P} X. \quad (3.62)$$

Obszar pozytywny zbioru X jest tożsamy z jego dolną aproksymacją.

DEFINICJA 3.8

\tilde{P} -brzegowy obszar zbioru X definiujemy jako

$$\text{Bn}_{\tilde{P}}(X) = \tilde{P} X \setminus \underline{P} X. \quad (3.63)$$

Przykład 3.14

Korzystając bezpośrednio z definicji 3.8, znajdziemy obszar brzegowy zbiorów X_F , X_N i X_O zdefiniowanych w przykładzie 3.8. Zrealizujemy to, wyznaczając różnicę zbiorów określonych w przykładach 3.11 i 3.8. Otrzymujemy zatem

$$\begin{aligned} \text{Bn}_{\tilde{C}}(X_F) &= \overline{\tilde{C}}X_F \setminus \underline{\tilde{C}}X_F \\ &= \{x_3, x_4, x_5, x_8\} \setminus \{x_3, x_8\} = \{x_4, x_5\}, \end{aligned} \quad (3.64)$$

$$\text{Bn}_{\tilde{C}}(X_N) = \overline{\tilde{C}}X_N \setminus \underline{\tilde{C}}X_N = \emptyset, \quad (3.65)$$

$$\text{Bn}_{\tilde{C}}(X_O) = \overline{\tilde{C}}X_O \setminus \underline{\tilde{C}}X_O = \{x_4, x_5\}. \quad (3.66)$$

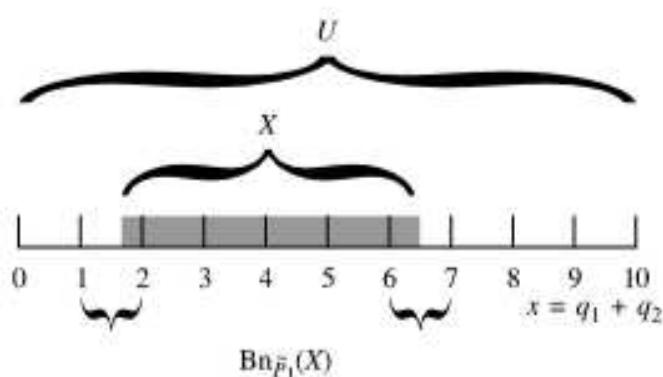
Przykład 3.15

Określmy obszar brzegowy zbioru X zdefiniowanego w przykładzie 3.9 dla zbioru cech P_1 oraz P_2 . W pierwszym przypadku mamy

$$\begin{aligned} \text{Bn}_{\tilde{P}_1}(X) &= \overline{\tilde{P}_1}X \setminus \underline{\tilde{P}_1}X \\ &= [1; 7) \setminus [2; 6) = [1; 2) \cup [6; 7), \end{aligned} \quad (3.67)$$

co ilustruje rysunek 3.10. W przypadku drugim

$$\begin{aligned} \text{Bn}_{\tilde{P}_2}(X) &= \overline{\tilde{P}_2}X \setminus \underline{\tilde{P}_2}X \\ &= U \setminus \emptyset = U. \end{aligned} \quad (3.68)$$



Rys. 3.10. Obszar brzegowy zbioru w jednowymiarowej przestrzeni rozważań

Przykład 3.16

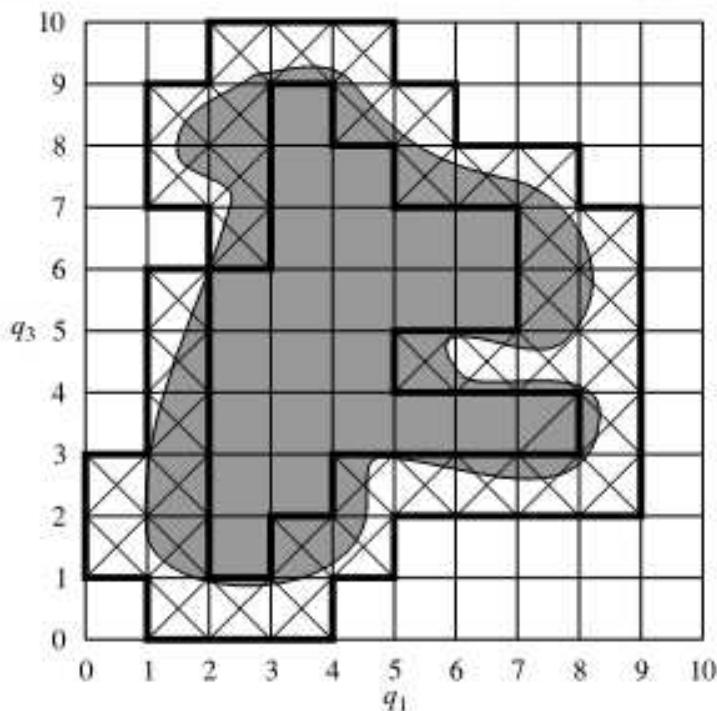
Na rysunku 3.11 zakreślone są klasy abstrakcji wchodzące w skład obszaru brzegowego $\text{Bn}_{\tilde{P}}(X)$, określonego w przestrzeni U zdefiniowanej wzorem (3.13).

DEFINICJA 3.9

\tilde{P} -negatywny obszar zbioru X definiujemy jako

$$\text{Neg}_{\tilde{P}}(X) = U \setminus \overline{\tilde{P}}X. \quad (3.69)$$

Negatywnym obszarem zbioru X jest zbiór tych obiektów $x \in U$, o których na podstawie wartości cech P możemy z całą pewnością powiedzieć, że nie są elementami zbioru X .



Rys. 3.11. Obszar brzegowy zbioru w dwuwy- miarowej przestrzeni rozwa ża n

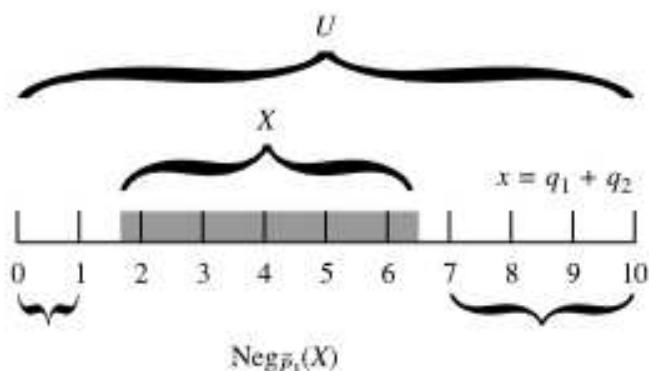
Przykład 3.17

Zgodnie z definicja 3.9 określmy obszary negatywne zbiorów X_F , X_N i X_O rozwa żanych w przykładzie 3.8. Wyznaczając dopełnienie zbiorów określonych w przykładzie 3.11 do przestrzeni U , otrzymujemy

$$\text{Neg}_{\tilde{C}}(X_F) = U \setminus \overline{\tilde{C}}X = \{x_1, x_2, x_6, x_7, x_9, x_{10}\}, \quad (3.70)$$

$$\text{Neg}_{\tilde{C}}(X_N) = U \setminus \overline{\tilde{C}}X = \{x_1, x_3, x_4, x_5, x_6, x_7, x_8\}, \quad (3.71)$$

$$\text{Neg}_{\tilde{C}}(X_O) = U \setminus \overline{\tilde{C}}X = \{x_2, x_3, x_8, x_9, x_{10}\}. \quad (3.72)$$



Rys. 3.12. Obszar negatywny zbioru w jednowymiarowej przestrzeni rozwa ża n

Przykład 3.18

Określmy obszar brzegowy zbioru X zdefiniowanego w przykładzie 3.9 dla zbioru cech P_1 oraz P_2 . W pierwszym przypadku mamy

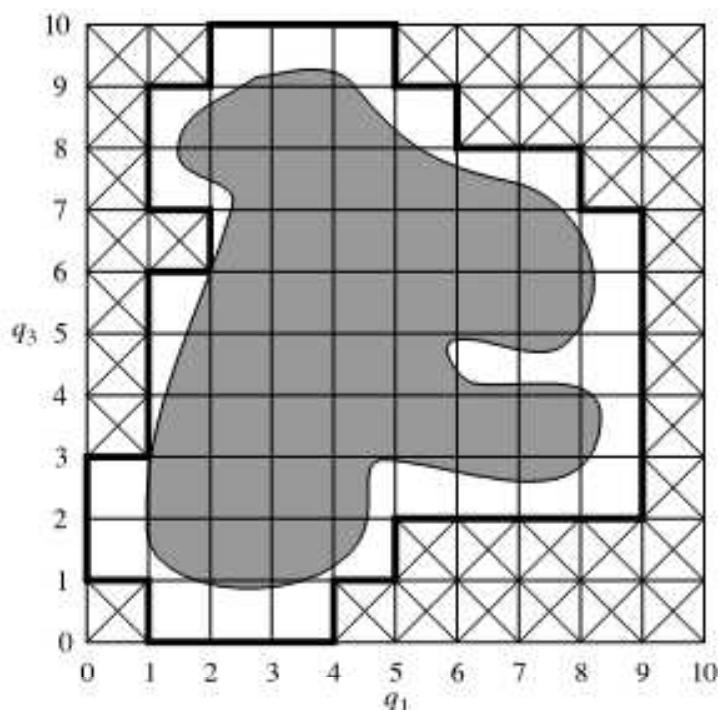
$$\begin{aligned} \text{Neg}_{\tilde{P}_1}(X) &= U \setminus \overline{\tilde{P}_1}X \\ &= U \setminus [1; 7) = [0; 1) \cup [7; 10), \end{aligned} \quad (3.73)$$

co ilustruje rysunek 3.12. W drugim przypadku mamy

$$\begin{aligned}\text{Neg}_{\tilde{P}_2}(X) &= U \setminus \overline{\tilde{P}_2 X} \\ &= U \setminus U = \emptyset.\end{aligned}\quad (3.74)$$

Przykład 3.19

Na rysunku 3.13 zakreślone są klasy abstrakcji wchodzące w skład obszaru negatywnego $\text{Neg}_{\tilde{P}}(X)$, określonego w przestrzeni U zdefiniowanej wzorem (3.13).



Rys. 3.13. Obszar negatywny zbioru w dwuwymiarowej przestrzeni rozważań

DEFINICJA 3.10

Zbiór X nazywamy \tilde{P} -dokładnym, jeżeli zachodzi równość jego dolnej i górnej aproksymacji

$$\underline{\tilde{P}}X = \overline{\tilde{P}}X \quad (3.75)$$

lub \tilde{P} -przybliżonym w przeciwnym razie

$$\underline{\tilde{P}}X \neq \overline{\tilde{P}}X. \quad (3.76)$$

DEFINICJA 3.11

Zbiór X nazywamy:

a) w przybliżeniu \tilde{P} -definiowalnym, jeżeli

$$\begin{cases} \underline{\tilde{P}}X \neq \emptyset \\ \overline{\tilde{P}}X \neq U, \end{cases} \quad (3.77)$$

b) wewnętrznie \tilde{P} -niedefiniowalnym, jeżeli

$$\begin{cases} \underline{\tilde{P}}X = \emptyset \\ \overline{\tilde{P}}X \neq U, \end{cases} \quad (3.78)$$

c) zewnętrznie \tilde{P} -niedefiniowalnym, jeżeli

$$\begin{cases} \underline{\tilde{P}}X \neq \emptyset \\ \overline{\tilde{P}}X = U, \end{cases} \quad (3.79)$$

d) całkowicie \tilde{P} -niedefiniowalnym, jeżeli

$$\begin{cases} \underline{\tilde{P}}X = \emptyset \\ \overline{\tilde{P}}X = U. \end{cases} \quad (3.80)$$

Przykład 3.20

Porównując dolne i górne aproksymacje zbiorów X_F , X_N i X_O , określone w przykładach 3.8 i 3.11, łatwo zauważyc, że jedynie zbiór X_N spełnia definicję 3.10 i jest \tilde{C} -dokładny. Zbiory X_F i X_O spełniają równanie (3.77) w definicji 3.11 i są zbiorami w przybliżeniu \tilde{C} -definiowalnymi, jak również \tilde{C} -przybliżonymi zgodnie z definicją 3.10.

Przykład 3.21

Porównując dolne i górne aproksymacje zbioru X zdefiniowanego w przykładzie 3.9, określone w przykładach 3.9 i 3.12, możemy stwierdzić, że zbiór ten jest zbiorem zarówno \tilde{P}_1 -, jak i \tilde{P}_2 -przybliżonym (definicja 3.10). Ponadto, z definicji 3.11 wynika, że jest on w przybliżeniu \tilde{P}_1 -definiowalny, a zarazem całkowicie \tilde{P}_2 -niedefiniowalny.

Przykład 3.22

Analizując rysunki 3.7 i 3.9, można stwierdzić, że zbiór X , zdefiniowany w przykładzie 3.10 na rysunku 3.7, jest zbiorem \tilde{P} -przybliżonym i zarazem w przybliżeniu \tilde{P} -definiowalnym.

DEFINICJA 3.12

Wartość wyrażoną wzorem

$$\mu_{\tilde{P}}(X) = \frac{\overline{\overline{\tilde{P}}X}}{\overline{\overline{\overline{\tilde{P}}X}}} \quad (3.81)$$

nazywamy \tilde{P} -dokładnością aproksymacji zbioru X . Symbolem $\overline{\overline{A}}$ oznaczono miarę zbioru A . W przypadku zbiorów skończonych jako miarę możemy przyjąć moc zbioru, w przypadku zbiorów ciągłych ograniczonych możemy posłużyć się takimi miarami, jak długość przedziału, pole powierzchni, objętość itd.

Przykład 3.23

Wyznaczmy \tilde{C} -dokładność zbiorów X_F , X_N i X_O , zdefiniowanych w przykładzie 3.8. Na mocy wzoru (3.81) mamy

$$\mu_{\tilde{C}}(X_F) = \frac{\overline{\overline{\tilde{C}}X_F}}{\overline{\overline{\overline{\tilde{C}}X_F}}} = \frac{2}{4} = 0,5, \quad (3.82)$$

$$\mu_{\tilde{C}}(X_N) = \frac{\overline{\overline{\tilde{C}}X_N}}{\overline{\overline{\overline{\tilde{C}}X_N}}} = \frac{3}{3} = 1, \quad (3.83)$$

$$\mu_{\tilde{C}}(X_O) = \frac{\overline{\overline{\tilde{C}X_O}}}{\overline{\overline{\overline{\tilde{C}X_O}}}} = \frac{3}{5} = 0,6. \quad (3.84)$$

Jak widzimy, \tilde{C} -dokładność aproksymacji zbioru X_N wynosi 1, co potwierdza wcześniejsze spostrzeżenie, że zbiór ten jest \tilde{C} -dokładny. Inaczej mówiąc, jest on jednoznacznie zdefiniowany przez cechy należące do zbioru C danego wzorem (3.23).

Przykład 3.24

W przypadku ciągłych przestrzeni rozważań możemy określić \tilde{C} -dokładność, korzystając z długości odpowiednich przedziałów. Zatem dla zbioru X , zdefiniowanego w przykładzie 3.9, mamy

$$\mu_{\tilde{P}_1}(X) = \frac{\overline{\overline{\tilde{P}_1 X}}}{\overline{\overline{\overline{\tilde{P}_1 X}}}} = \frac{2}{3}, \quad (3.85)$$

$$\mu_{\tilde{P}_2}(X) = \frac{\overline{\overline{\tilde{P}_2 X}}}{\overline{\overline{\overline{\tilde{P}_2 X}}}} = 0. \quad (3.86)$$

Przykład 3.25

W przypadku zbioru X , zdefiniowanego w przykładzie 3.10, \tilde{P} -dokładność aproksymacji wyznaczmy, korzystając z miary, jaką jest pole powierzchni. Na podstawie rysunków 3.7 i 3.9 mamy

$$\mu_{\tilde{P}}(X) = \frac{\overline{\overline{\overline{\tilde{P}X}}}}{\overline{\overline{\overline{\overline{\tilde{P}X}}}}} = \frac{8}{21}. \quad (3.87)$$

3.4. Aproksymacja rodziny zbiorów

Definicje 3.5, 3.6, 3.7, 3.8, 3.9 oraz 3.12 można bez przeszkód uogólnić dla pewnej rodziny zbiorów przestrzeni U . Wspomnianą rodzinę zbiorów oznaczmy przez $X = \{X_1, X_2, \dots, X_n\}$.

DEFINICJA 3.13

\tilde{P} -dolną aproksymacją rodziny zbiorów X nazywamy zbiór $\underline{\tilde{P}X}$ opisany następująco:

$$\underline{\tilde{P}X} = \{\underline{\tilde{P}X}_1, \underline{\tilde{P}X}_2, \dots, \underline{\tilde{P}X}_n\}. \quad (3.88)$$

Przykład 3.26

Niech elementami rodziny zbiorów X będą zbiory X_F , X_N oraz X_O , zdefiniowane w przykładzie 3.8. Zapiszemy to następująco:

$$\begin{aligned} X &= \{X_F, X_N, X_O\} \\ &= \{\{x_3, x_4, x_8\}, \{x_2, x_9, x_{10}\}, \{x_1, x_5, x_6, x_7\}\}. \end{aligned} \quad (3.89)$$

Korzystając ze zbiorów wyznaczonych w przykładzie 3.8, możemy zgodnie z definicją 3.13 zapisać

$$\begin{aligned}\widetilde{C}X &= \{\widetilde{C}X_F, \widetilde{C}X_N, \widetilde{C}X_O\} \\ &= \{\{x_3, x_8\}, \{x_2, x_9, x_{10}\}, \{x_1, x_6, x_7\}\}.\end{aligned}\quad (3.90)$$

DEFINICJA 3.14

\widetilde{P} -górną aproksymacją rodziny zbiorów X nazywamy zbiór $\widetilde{P}X$ opisany następująco:

$$\widetilde{P}X = \{\widetilde{P}X_1, \widetilde{P}X_2, \dots, \widetilde{P}X_n\}. \quad (3.91)$$

Przykład 3.27

Zgodnie z definicją 3.14, z wykorzystaniem zbiorów wyznaczonych w przykładzie 3.11, \widetilde{C} -górną aproksymacją rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, wyniesie

$$\begin{aligned}\widetilde{C}X &= \{\widetilde{C}X_F, \widetilde{C}X_N, \widetilde{C}X_O\} \\ &= \{\{x_3, x_4, x_5, x_8\}, \{x_2, x_9, x_{10}\}, \{x_1, x_4, x_5, x_6, x_7\}\}.\end{aligned}\quad (3.92)$$

DEFINICJA 3.15

\widetilde{P} -pozytywny obszar rodziny zbiorów X definiujemy jako

$$\text{Pos}_{\widetilde{P}}(X) = \bigcup_{X_i \in X} \text{Pos}_{\widetilde{P}}(X_i). \quad (3.93)$$

Przykład 3.28

\widetilde{C} -pozytywny obszar rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, możemy wyznaczyć następująco:

$$\begin{aligned}\text{Pos}_{\widetilde{C}}(X) &= \text{Pos}_{\widetilde{C}}(X_F) \cup \text{Pos}_{\widetilde{C}}(X_N) \cup \text{Pos}_{\widetilde{C}}(X_O) \\ &= \{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\}.\end{aligned}\quad (3.94)$$

Jak wynika z przykładu, pojęcie obszaru pozytywnego rodziny zbiorów nie jest tożsame z pojęciem jej dolnej aproksymacji — w przeciwnieństwie do pojęć obszaru pozytywnego i dolnej aproksymacji zbiorów.

DEFINICJA 3.16

\widetilde{P} -brzegowy obszar rodziny zbiorów X definiujemy jako

$$\text{Bn}_{\widetilde{P}}(X) = \bigcup_{X_i \in X} \text{Bn}_{\widetilde{P}}(X_i). \quad (3.95)$$

Przykład 3.29

Zgodnie z definicją 3.16, \widetilde{C} -brzegowy obszar rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, przybierze postać

$$\begin{aligned}\text{Bn}_{\widetilde{C}}(X) &= \text{Bn}_{\widetilde{C}}(X_F) \cup \text{Bn}_{\widetilde{C}}(X_N) \cup \text{Bn}_{\widetilde{C}}(X_O) \\ &= \{x_4, x_5\}.\end{aligned}\quad (3.96)$$

DEFINICJA 3.17

\tilde{P} -negatywny obszar rodziny zbiorów X definiujemy jako

$$\text{Neg}_{\tilde{P}}(X) = U \setminus \bigcup_{X_i \in X} \overline{\tilde{P}} X_i. \quad (3.97)$$

Przykład 3.30

\tilde{C} -negatywny obszar rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, zgodnie z definicją 3.17 przybiera postać

$$\text{Neg}_{\tilde{C}}(X) = U \setminus \bigcup_{X_i \in X} \overline{\tilde{C}} X_i = \emptyset. \quad (3.98)$$

DEFINICJA 3.18

\tilde{P} -jakość aproksymacji rodziny zbiorów X określona jest wyrażeniem

$$\gamma_{\tilde{P}}(X) = \frac{\overline{\overline{\text{Pos}_{\tilde{P}}(X)}}}{\overline{\overline{U}}}. \quad (3.99)$$

Przykład 3.31

\tilde{C} -jakość aproksymacji rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, wynosi

$$\gamma_{\tilde{C}}(X) = \frac{\overline{\overline{\text{Pos}_{\tilde{C}}(X)}}}{\overline{\overline{U}}} = \frac{8}{10}. \quad (3.100)$$

DEFINICJA 3.19

\tilde{P} -dokładność aproksymacji rodziny zbiorów X określamy przez

$$\beta_{\tilde{P}}(X) = \frac{\overline{\overline{\text{Pos}_{\tilde{P}}(X)}}}{\sum_{X_i \in X} \overline{\overline{\tilde{P}} X_i}}. \quad (3.101)$$

Przykład 3.32

Korzystając z definicji oraz z zapisu (3.92) i (3.94), łatwo sprawdzić, że \tilde{C} -dokładność aproksymacji rodziny zbiorów X , zdefiniowanej w przykładzie 3.26, wynosi

$$\beta_{\tilde{C}}(X) = \frac{\overline{\overline{\text{Pos}_{\tilde{C}}(X)}}}{\sum_{X_i \in X} \overline{\overline{\tilde{C}} X_i}} = \frac{8}{4+3+5} = \frac{2}{3}. \quad (3.102)$$

3.5. Analiza tablic decyzyjnych

W teorii zbiorów przybliżonych wprowadza się pojęcie zależności między cechami (atrybutami) systemu informacyjnego. Dzięki temu można sprawdzić, czy do jednoznacznego opisu obiektu należącego do zbioru U konieczna jest znajomość wartości wszystkich jego cech.

DEFINICJA 3.20

Stopień zależności zbioru atrybutów P_2 od zbioru atrybutów P_1 , gdzie $P_1, P_2 \subseteq Q$, jest określony następująco:

$$k = \gamma_{\tilde{P}_1}(P_2^*), \quad (3.103)$$

przy czym $\gamma_{\tilde{C}}(D^*)$ wyznacza się zgodnie z definicją 3.18.

Zapis $P_1 \xrightarrow{k} P_2$ oznacza, że zbiór atrybutów P_2 zależy od zbioru atrybutów P_1 w stopniu $k < 1$. W przypadku gdy $k = 1$, piszemy po prostu $P_1 \rightarrow P_2$.

Pojęcie stopnia zależności atrybutów wykorzystuje się do określenia poprawności budowy tablicy decyzyjnej (definicja 3.2).

DEFINICJA 3.21

Reguły tablicy decyzyjnej nazywamy *deterministycznymi*, jeżeli dla każdej pary reguł $l_a \neq l_b$ z równości wartości wszystkich atrybutów warunkowych C wynika równość wartości atrybutów decyzyjnych D , tzn.

$$\forall_{l_a, l_b=1, \dots, N} : \forall_{c \in C} f_{l_a}(c) = f_{l_b}(c) \rightarrow \forall_{d \in D} f_{l_a}(d) = f_{l_b}(d). \quad (3.104)$$

Jeżeli dla pewnej pary reguł $l_a \neq l_b$ powyższy warunek nie jest spełniony, tzn. z równości wartości wszystkich atrybutów warunkowych C nie wynika równość wartości atrybutów decyzyjnych D , reguły te nazywamy *niedeterministycznymi*.

$$\exists_{l_a, l_b} : \forall_{c \in C} f_{l_a}(c) = f_{l_b}(c) \rightarrow \exists_{d \in D} f_{l_a}(d) \neq f_{l_b}(d). \quad (3.105)$$

Tablica decyzyjna (definicja 3.2) jest *dobrze określona*, jeśli jej wszystkie reguły są deterministyczne. W przeciwnym przypadku mówimy, że jest *źle określona*.

Zauważmy, że tablica decyzyjna posiadająca zbiór atrybutów warunkowych C oraz zbiór atrybutów decyzyjnych D jest dobrze określona, jeśli zbiór atrybutów decyzyjnych zależy od zbiuru atrybutów warunkowych w stopniu równym 1 ($C \rightarrow D$), czyli

$$\gamma_{\tilde{C}}(D^*) = 1. \quad (3.106)$$

Źródłem złego określenia tablicy decyzyjnej jest istnienie w niej tzw. reguł niedeterministycznych. Źle określoną tablicę decyzyjną można „naprawić” przez usunięcie reguł niedeterministycznych lub rozszerzenie zbiuru atrybutów warunkowych C .

Przykład 3.33

Zbadajmy stopień zależności zbioru atrybutów D od zbioru atrybutów C zdefiniowanych w przykładzie 3.4. Zgodnie z definicją 3.20 mamy

$$k = \gamma_{\tilde{C}}(D^*) = \frac{\overline{\text{Pos}_{\tilde{C}}(D^*)}}{\overline{\overline{U}}}, \quad (3.107)$$

Zauważmy, że klasy abstrakcji relacji D -nieroróżnialności to zdefiniowane w przykładzie 3.8 zbiory X_F , X_N i X_O . Zatem

$$D^* = \{X_F, X_N, X_O\}. \quad (3.108)$$

Na podstawie definicji 3.15, wykorzystując wyznaczone w przykładzie 3.8 dolne aproksymacje zbiorów X_F , X_N i X_O , otrzymujemy

$$\begin{aligned}\text{Pos}_{\tilde{C}}(D^*) &= \text{Pos}_{\tilde{C}}(X_F) \cup \text{Pos}_{\tilde{C}}(X_N) \cup \text{Pos}_{\tilde{C}}(X_O) = \underline{\tilde{C}}X_F \cup \underline{\tilde{C}}X_N \cup \underline{\tilde{C}}X_O \\ &= \{x_3, x_8\} \cup \{x_2, x_9, x_{10}\} \cup \{x_1, x_6, x_7\} \\ &= \{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\}.\end{aligned}\quad (3.109)$$

Podstawiając zależność (3.109) do wzoru (3.107), otrzymujemy wynik

$$k = \frac{\overline{\{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\}}}{\overline{\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}}} = \frac{8}{10}. \quad (3.110)$$

Możemy zatem stwierdzić, że zbiór atrybutów D zależy od zbioru atrybutów C w stopniu $k = 0,8$, co zapisujemy $C \xrightarrow{0.8} D$. Otrzymana wartość $k < 1$ informuje nas, że tablica decyzyjna dana w przykładzie 3.4 jest źle określona. Na podstawie zbioru atrybutów warunkowych C nie można jednoznacznie wnioskować o przynależności obiektów przestrzeni U do poszczególnych zbiorów X_F , X_N i X_O , będących klasami abstrakcji relacji D -nieroróżnicialności.

Właściciel komisu z przykładu 3.1 powinien rozszerzyć wykorzystywany zbiór atrybutów warunkowych C , jeśli chce na ich podstawie jednoznacznie wnioskować o marce pojazdu.

Regułami niedeterministycznymi w tablicy decyzyjnej opisanej w przykładzie 3.4 są reguły 4 i 5. Ich usunięcie przekształca tablicę decyzyjną źle określoną (tab. 3.3) w tablicę decyzyjną dobrze określoną (tab. 3.4).

Tabela 3.4. Tablica decyzyjna dobrze określona (po usunięciu reguł niedeterministycznych)

Reguła (l)	Liczba drzwi (c_1)	Moc silnika (c_2)	Kolor (c_3)	Marka (d_1)
1	2	60	niebieski	Opel
2	2	100	czarny	Nissan
3	2	200	czarny	Ferrari
6	3	100	czerwony	Opel
7	3	100	czerwony	Opel
8	3	200	czarny	Ferrari
9	4	100	niebieski	Nissan
10	4	100	niebieski	Nissan

Dla tak zdefiniowanej tablicy decyzyjnej łatwo sprawdzić, że

$$\gamma_{\tilde{C}}(D^*) = \frac{\overline{\{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\}}}{\overline{\{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\}}} = 1. \quad (3.111)$$

Zatem jest ona dobrze określona.

Drugą metodą stworzenia dobrze określonej tablicy jest rozszerzenie zbioru atrybutów warunkowych. Właściciel komisu postanowił dodać do dotychczas rozważanych cech rodzaj stosowanego paliwa, rodzaj tapicerki oraz felg. Nowy zbiór atrybutów warunkowych przybiera postać

$$\begin{aligned} C &= \{c_1, c_2, c_3, c_4, c_5, c_6\} \\ &= \{\text{liczba drzwi, moc silnika, kolor, paliwo, tapicerka, felgi}\}. \end{aligned} \quad (3.112)$$

Dziedziny nowych cech to

$$V_{c_4} = \{\text{ON, ET, gaz}\}, \quad (3.113)$$

$$V_{c_5} = \{\text{tkanina, skóra}\}, \quad (3.114)$$

$$V_{c_6} = \{\text{stal, aluminium}\}. \quad (3.115)$$

Tabela 3.5 przedstawia tablicę decyzyjną uzupełnioną o nowe atrybuty i ich wartości.

Tabela 3.5. Tablica decyzyjna dobrze określona (po dodaniu atrybutów)

Reguła (l)	Liczba drzwi (c_1)	Moc silnika (c_2)	Kolor (c_3)	Paliwo (c_4)	Tapicerka (c_5)	Felgi (c_6)	Marka (d_1)
1	2	60	niebieski	ET	tkanina	stal	Opel
2	2	100	czarny	ON	tkanina	stal	Nissan
3	2	200	czarny	ET	skóra	Al	Ferrari
4	2	200	czerwony	ET	skóra	Al	Ferrari
5	2	200	czerwony	ET	tkanina	stal	Opel
6	3	100	czerwony	ON	skóra	stal	Opel
7	3	100	czerwony	gaz	tkanina	stal	Opel
8	3	200	czarny	ET	skóra	Al	Ferrari
9	4	100	niebieski	gaz	tkanina	stal	Nissan
10	4	100	niebieski	ON	tkanina	Al	Nissan

Wyznaczmy dla tablicy decyzyjnej zdefiniowanej w tabeli 3.5 \tilde{C} -jakość oraz \tilde{C} -dokładność aproksymacji rodziny zbiorów D^* . Pierwszym krokiem będzie wyznaczenie rodziny klas abstrakcji relacji \tilde{C} w przestrzeni U . Każdy element przestrzeni U ma przynajmniej jedną różną wartość cechy, więc

$$C^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\}. \quad (3.116)$$

Zatem obszary pozytywne rodziny zbiorów B^* będą następujące:

$$\text{Pos}_{\tilde{C}}(X_F) = \underline{\tilde{C}} X_F = \{x_3\} \cup \{x_4\} \cup \{x_8\} = \{x_3, x_4, x_8\} = X_F, \quad (3.117)$$

$$\text{Pos}_{\tilde{C}}(X_N) = \underline{\tilde{C}} X_N = \{x_2\} \cup \{x_9\} \cup \{x_{10}\} = \{x_2, x_9, x_{10}\} = X_N, \quad (3.118)$$

$$\text{Pos}_{\tilde{C}}(X_O) = \underline{\tilde{C}} X_O = \{x_1\} \cup \{x_5\} \cup \{x_6\} \cup \{x_7\} = \{x_1, x_5, x_6, x_7\} = X_O. \quad (3.119)$$

Podobnie można określić \tilde{C} -górnne aproksymacje tych zbiorów

$$\overline{\tilde{C}} X_F = \{x_3\} \cup \{x_4\} \cup \{x_8\} = \{x_3, x_4, x_8\} = X_F, \quad (3.120)$$

$$\overline{\tilde{C}}X_N = \{x_2\} \cup \{x_9\} \cup \{x_{10}\} = \{x_2, x_9, x_{10}\} = X_N, \quad (3.121)$$

$$\overline{\tilde{C}}X_O = \{x_1\} \cup \{x_5\} \cup \{x_6\} \cup \{x_7\} = \{x_1, x_5, x_6, x_7\} = X_O. \quad (3.122)$$

\tilde{C} -pozytywny obszar rodziny zbiorów D^* przybierze postać

$$\begin{aligned} \text{Pos}_{\tilde{C}}(D^*) &= \{x_3, x_4, x_8\} \cup \{x_2, x_9, x_{10}\} \cup \{x_1, x_5, x_6, x_7\} \\ &= \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\} = U. \end{aligned} \quad (3.123)$$

Teraz można już bezpośrednio, korzystając z definicji 3.18 i 3.19, wyznaczyć

$$\gamma_{\tilde{C}}(D^*) = \frac{\overline{\text{Pos}_{\tilde{C}}(D^*)}}{\overline{\overline{U}}} = \frac{\overline{\overline{U}}}{\overline{\overline{U}}} = \frac{10}{10} = 1, \quad (3.124)$$

$$\beta_{\tilde{C}}(D^*) = \frac{\overline{\overline{\text{Pos}_{\tilde{C}}(D^*)}}}{\sum_{X_i \in D^*} \overline{\overline{\tilde{C}X_i}}} = \frac{10}{3 + 3 + 4} = 1. \quad (3.125)$$

Na tej podstawie można jednoznacznie stwierdzić, że tablica decyzyjna przedstawiona w tabeli 3.5 jest dobrze określona.

DEFINICJA 3.22

Zbiór atrybutów $P_1 \subseteq Q$ jest *niezależny* w danym systemie informacyjnym, jeżeli dla każdego $P_2 \subset P_1$ zachodzi $\tilde{P}_1 \neq \tilde{P}_2$. W przeciwnym przypadku zbiór P_1 jest *zależny*.

Przykład 3.34

Rozważmy dane zawarte w tablicy decyzyjnej 3.5. Łatwo zauważyc, że zbiór C opisany wzorem (3.112) jest zbiorem zależnym. Przykładowe podzbiory zbioru C , $C_1 = \{c_1, c_2, c_3, c_5, c_6\}$ i $C_2 = \{c_1, c_2, c_3, c_4, c_5\}$, generują taki iloraz przestrzeni U , jak zbiór C (por. (3.116)). Zbiory C_1 i C_2 również są zależne, gdyż zbiory $C_3 = \{c_1, c_3, c_5, c_6\} \subset C_1$ i $C_4 = \{c_1, c_3, c_4, c_5\} \subset C_2$ generują także iloraz przestrzeni U opisany wzorem (3.116). Natomiast zbiory C_3 i C_4 są zbiorami niezależnymi.

DEFINICJA 3.23

Zbiór atrybutów $P_1 \subseteq Q$ jest *niezależny ze względu* na zbiór atrybutów $P_2 \subseteq Q$ (P_2 -*niezależny*), jeśli dla każdego $P_3 \subset P_1$ zachodzi

$$\text{Pos}_{\tilde{P}_1}(P_2^*) \neq \text{Pos}_{\tilde{P}_3}(P_2^*). \quad (3.126)$$

W przeciwnym przypadku zbiór P_1 jest P_2 -*zależny*.

Przykład 3.35

Zbiór C_3 jest zbiorem zależnym w danym systemie informacyjnym (definicja 3.22). Z punktu widzenia definicji 3.23 jest to zbiór D -zależny. Wykażemy, że zbiór C_3 wraz ze swoim podzbiorem $C_5 = \{c_1, c_3, c_6\}$ nie spełniają warunku (3.126), tzn.

$$\text{Pos}_{\tilde{C}_3}(D^*) \neq \text{Pos}_{\tilde{C}_5}(D^*). \quad (3.127)$$

Zauważmy, że

$$\begin{aligned} \text{Pos}_{\tilde{C}_3}(D^*) &= \text{Pos}_{\tilde{C}_3}(X_F) \cup \text{Pos}_{\tilde{C}_3}(X_N) \cup \text{Pos}_{\tilde{C}_3}(X_O) \\ &= X_F \cup X_N \cup X_O = U \end{aligned} \quad (3.128)$$

oraz

$$\begin{aligned}\text{Pos}_{\tilde{C}_s}(D^*) &= \text{Pos}_{\tilde{C}_s}(X_F) \cup \text{Pos}_{\tilde{C}_s}(X_N) \cup \text{Pos}_{\tilde{C}_s}(X_O) \\ &= X_F \cup X_N \cup X_O = U,\end{aligned}\quad (3.129)$$

zatem

$$\text{Pos}_{\tilde{C}_s}(D^*) = \text{Pos}_{\tilde{C}_s}(D^*). \quad (3.130)$$

DEFINICJA 3.24

Reduktem zbioru atrybutów $P_1 \subseteq Q$ nazywamy każdy niezależny zbiór $P_2 \subset P_1$, dla którego $\tilde{P}_2 = \tilde{P}_1$.

DEFINICJA 3.25

Reduktem względnym zbioru atrybutów $P_1 \subseteq Q$ ze względu na P_2 (tzw. P_2 -reduktem) nazywamy każdy P_2 -niezależny zbiór $P_3 \subset P_1$, dla którego $\tilde{P}_3 = \tilde{P}_1$.

Przykład 3.36

Wracając do rozważań przeprowadzonych w przykładach 3.34 i 3.35, możemy zauważyć, że przedstawione tam zbiory C_3 i C_4 są reduktami zbioru C , natomiast zbiór C_5 jest jego D -reduktem.

DEFINICJA 3.26

Atrybut $p \in P_1$ jest *nieusuwalny* z P_1 , jeżeli dla $P_2 = P_1 \setminus \{p\}$ zachodzi $\tilde{P}_2 \neq \tilde{P}_1$. W przeciwnym przypadku atrybut p jest *zbędny*.

Przykład 3.37

Korzystając z definicji 3.26, sprawdzimy nieusuwalność poszczególnych atrybutów $c \in C$ w systemie informacyjnym tworzącym tablicę decyzyjną 3.5. Łatwo sprawdzić, że

$$C^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\}, \quad (3.131)$$

$$(C \setminus \{c_1\})^* = \{\{x_1\}, \{x_2\}, \{x_3, x_8\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_9\}, \{x_{10}\}\} \neq C^*, \quad (3.132)$$

$$(C \setminus \{c_2\})^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\} = C^*, \quad (3.133)$$

$$(C \setminus \{c_3\})^* = \{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\} \neq C^*, \quad (3.134)$$

$$(C \setminus \{c_4\})^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\} = C^*, \quad (3.135)$$

$$(C \setminus \{c_5\})^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\} = C^*, \quad (3.136)$$

$$(C \setminus \{c_6\})^* = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\} = C^*. \quad (3.137)$$

Jak widać, atrybuty c_1 i c_3 są nieusuwalne, natomiast atrybuty c_2, c_4, c_5 i c_6 są zbędne.

DEFINICJA 3.27

Zbiór wszystkich atrybutów nieusuwalnych ze zbioru P nazywa się *rdzeniem* P , co zapisujemy następująco:

$$\text{CORE}(P) = \{p \in P : \tilde{P}' \neq \tilde{P}, P' = P \setminus \{p\}\}. \quad (3.138)$$

Przykład 3.38

Korzystając z rozważań przeprowadzonych w przykładzie 3.37, można wyznaczyć rdzeń zbioru atrybutów C jako

$$\text{CORE}(C) = \{c_1, c_3\}. \quad (3.139)$$

DEFINICJA 3.28

Znormalizowany współczynnik istotności podzbioru zbioru atrybutów warunkowych $C' \subset C$ wyrażony jest przez

$$\sigma_{(C,D)}(C') = \frac{\gamma_{\bar{C}}(D^*) - \gamma_{\bar{C}''}(D^*)}{\gamma_{\bar{C}}(D^*)}, \quad (3.140)$$

gdzie $C'' = C \setminus C'$. Oczywiście w szczególnym przypadku zbiór C' może być jednoelementowy, wówczas współczynnik (3.140) będzie wyrażał istotność jednego atrybutu warunkowego.

Współczynnik istotności odgrywa ważną rolę w analizie reguł tablic decyzyjnych. Wartość zerowa otrzymana dla danego podzbioru atrybutów warunkowych C wskazuje, że podzbiór ten może być usunięty ze zbioru atrybutów warunkowych bez szkody dla aproksymacji rodziny zbiorów D^* .

Przykład 3.39

Wyznaczmy istotność przykładowego podzbioru zbioru atrybutów warunkowych C określonego zapisem (3.112). W przykładzie 3.33 pokazaliśmy (wzór (3.124)), że \bar{C} -jakość aproksymacji rodziny zbiorów D^* dla dobrze określonej tablicy decyzyjnej wynosi 1. Dla $C' = \{c_1\}$ mamy $C'' = \{c_2, c_3, c_4, c_5, c_6\}$ oraz

$$\begin{aligned} \gamma_{\bar{C}''}(D^*) &= \frac{\overline{\text{Pos}_{\bar{C}''}(D^*)}}{\overline{U}} \\ &= \frac{\overline{X_F \cup X_N \cup X_O}}{\overline{U}} = 1. \end{aligned} \quad (3.141)$$

Stąd

$$\sigma_{(C,D)}(\{c_1\}) = \frac{1 - 1}{1} = 0. \quad (3.142)$$

Zatem atrybut c_1 jest w danej tablicy decyzyjnej nieistotny, w związku z tym jego usunięcie nie wpłynie na jakość aproksymacji rodziny zbiorów D^* .

Dla $C' = \{c_4, c_5, c_6\}$ otrzymujemy $C'' = \{c_1, c_2, c_3\}$, a więc

$$\gamma_{\bar{C}''}(D^*) = \frac{\overline{\{x_3, x_8\} \cup \{x_2, x_9, x_{10}\} \cup \{x_1, x_6, x_7\}}}{\overline{U}} = \frac{8}{10}, \quad (3.143)$$

co po podstawieniu do wzoru (3.140) daje wartość

$$\sigma_{(C,D)}(\{c_4, c_5, c_6\}) = \frac{1 - 0,8}{1} = 0,2. \quad (3.144)$$

Z powyższych rozważań wynika, że dodane w przykładzie 3.33 (tab. 3.5) atrybuty c_4 , c_5 i c_6 mają niewielką istotność.

DEFINICJA 3.29

Dowolny podzbiór zbioru atrybutów warunkowych $C' \subset C$ nazywamy przybliżonym D -reduktem zbioru atrybutów C , a *błąd przybliżenia* tego reduktu wyznaczamy następująco:

$$\varepsilon_{(C,D)}(C') = \frac{\gamma_{\bar{C}}(D^*) - \gamma_{\bar{C}'}(D^*)}{\gamma_{\bar{C}}(D^*)}. \quad (3.145)$$

Przykład 3.40

Wyznaczmy błąd przybliżenia zbioru $C' = \{c_1, c_2, c_3\}$ będącego przybliżonym D -reduktem zbioru atrybutów C (tablica decyzyjna 3.5). Wykorzystując wynik (3.143), możemy zapisać

$$\varepsilon_{(C,D)}(\{c_1, c_2, c_3\}) = \frac{1 - 0,8}{1} = 0,2. \quad (3.146)$$

3.6. Zastosowanie programu LERS

Program LERS (ang. *Learning from Examples based on Rough Sets*) [67] jest dziełem firmy RS Systems. Jego zadaniem jest wygenerowanie bazy reguł na podstawie wprowadzonych przykładów oraz testowanie powstałej bądź niezależnie przygotowanej bazy reguł. Wprowadzone dane mogą być poddane wstępnej obróbce, m.in. przez usunięcie sprzeczności, eliminację lub uzupełnienie brakujących danych oraz kwantyzację wartości numerycznych.

W celu przedstawienia możliwości wykorzystania programu LERS rozważmy dwa przypadki analizy danych. Pierwszy — to rozważany już w przykładzie 3.1 przypadek komisu samochodowego. Drugi — to zagadnienie klasyfikacji kwiatów irysa, przykład często stosowany do ilustracji działania algorytmów inteligencji obliczeniowej.

Przykład 3.41 — Samochody na parkingu

Niech tablica decyzyjna opisująca komis samochodowy z przykładu 3.1 ma postać jak w tabeli 3.5. W celu klarowności prezentacji niniejszego przykładu została ona powtórnie przedstawiona w tabeli 3.6.

Tabela 3.6. Pierwotna tablica decyzyjna (przed redukcją)

Liczba drzwi (c ₁)	Moc silnika (c ₂)	Kolor (c ₃)	Paliwo (c ₄)	Tapicerka (c ₅)	Felgi (c ₆)	Marka (d ₁)
2	60	niebieski	ET	tkanina	stal	Opel
2	100	czarny	ON	tkanina	stal	Nissan
2	200	czarny	ET	skóra	alum	Ferrari
2	200	czerwony	ET	skóra	alum	Ferrari
2	200	czerwony	ET	tkanina	stal	Opel
3	100	czerwony	ON	skóra	stal	Opel
3	100	czerwony	gaz	tkanina	stal	Opel
3	200	czarny	ET	skóra	alum	Ferrari
4	100	niebieski	gaz	tkanina	stal	Nissan
4	100	niebieski	ON	tkanina	alum	Nissan

Aby dane z tabeli 3.6 wprowadzić do programu LERS, należy przygotować następujący plik:

< a a a a a a a d >							
[drzwi	moc	kolor	paliwo	tapicerka	felgi	marka]
2	60	niebieski	ET	tkanina	stal	Opel	
2	100	czarny	ON	tkanina	stal	Nissan	
2	200	czarny	ET	skóra	alum	Ferrari	
2	200	czerwony	ET	skóra	alum	Ferrari	
2	200	czerwony	ET	tkanina	stal	Opel	
3	100	czerwony	ON	skóra	stal	Opel	
3	100	czerwony	gaz	tkanina	stal	Opel	
3	200	czarny	ET	skóra	alum	Ferrari	
4	100	niebieski	gaz	tkanina	stal	Nissan	
4	100	niebieski	ON	tkanina	alum	Nissan	

W pierwszym wierszu pliku został zdefiniowany podział na atrybuty warunkowe (a) i atrybuty decyzyjne (d). W drugim wierszu figurują nazwy poszczególnych atrybutów. Na podstawie wprowadzonych danych program LERS wygenerował 5 reguł zawierających łącznie 8 warunków:

JEŻELI felgi jest stal **I** kolor jest czerwony **TO** marka jest Opel

JEŻELI moc jest 60 **TO** marka jest Opel

JEŻELI drzwi jest 4 **TO** marka jest Nissan

JEŻELI kolor jest czarny **I** paliwo jest ON **TO** marka jest Nissan

JEŻELI moc jest 200 **I** tapicerka jest skóra **TO** marka jest Ferrari

Proces generowania reguł można interpretować jako usunięcie z tablicy decyzyjnej zbędnych danych, co ilustruje tabela 3.7. Algorytm generacji reguł oraz usuwania zbędnych danych wykorzystuje teorię zbiorów przybliżonych.

Tabela 3.7. Tablica decyzyjna po usunięciu zbędnych danych

Liczba drzwi (c_1)	Moc silnika (c_2)	Kolor (c_3)	Paliwo (c_4)	Tapicerka (c_5)	Felgi (c_6)	Marka (d_1)
	60					Opel
		czarny	ON			Nissan
	200			skóra		Ferrari
	200			skóra		Ferrari
		czerwony			stal	Opel
		czerwony			stal	Opel
		czerwony			stal	Opel
	200			skóra		Ferrari
4						Nissan
4						Nissan

Usuwając z tabeli 3.7 powtarzające się wpisy, uzyskujemy tabelę 3.8, tożsamą z wygenerowanym zbiorem reguł.

Tabela 3.8. Tablica decyzyjna uzyskana po redukcji

Liczba drzwi (c_1)	Moc silnika (c_2)	Kolor (c_3)	Paliwo (c_4)	Tapicerka (c_5)	Felgi (c_6)	Marka (d_1)
		czerwony			stal	Opel
	60					Opel
4						Nissan
		czarny	ON			Nissan
	200			skóra		Ferrari

Przykład 3.42 — Klasyfikacja kwiatów irysa

Jak już wspomniano, zagadnienie klasyfikacji kwiatów irysa jest często stosowanym przykładem ilustrującym działanie różnego typu algorytmów inteligencji komputerowej. Zadanie polega na określeniu przynależności kwiatów do jednej z trzech klas: Setosa, Virginica i Versicolor. Decyzja jest podejmowana na podstawie wartości czterech atrybutów warunkowych opisujących wymiary (długość i szerokość) liścia i płatka rośliny.

Do dyspozycji mamy 150 próbek, w tym 147 unikalnych (nie powtarzających się). Do każdej z trzech klas należy 50 z nich. Tabela 3.9 przedstawia zakresy zmienności poszczególnych atrybutów.

Dane zostały podzielone na część uczącą i testową. Do części uczącej wybrano losowo po 40 próbek z każdej klasy, a pozostałe 30 próbek zostało wykorzystanych do utworzenia części testowej. Na podstawie zawartości ciągu uczącego przygotowano plik

Tabela 3.9. Zakresy zmienności atrybutów (klasyfikacja kwiatów irysa)

Atrybut	Zakres	Liczba unikalnych wartości
p1	(4,3; 7,9)	35
p2	(2,0; 4,4)	23
p3	(1,0; 6,9)	43
p4	(0,1; 2,5)	22
Irys	Setosa, Virginica, Versicolor	3

wejściowy dla programu LERS w postaci:

```

< a a a a a d >
[p1  p2  p3  p4  irys]
4.4  2.9  1.4  0.2  Setosa
4.8  3.0  1.4  0.1  Setosa
5.4  3.4  1.7  0.2  Setosa
...

```

Na podstawie wprowadzonych danych program LERS wygenerował 34 reguły zawierające razem 41 warunków:

- JEŻELI** p4 jest 0,2 **TO** irys jest Setosa
- JEŻELI** p4 jest 0,4 **TO** irys jest Setosa
- JEŻELI** p4 jest 0,3 **TO** irys jest Setosa
- JEŻELI** p4 jest 0,1 **TO** irys jest Setosa
- JEŻELI** p4 jest 0,5 **TO** irys jest Setosa
- JEŻELI** p4 jest 0,6 **TO** irys jest Setosa
- JEŻELI** p4 jest 1,3 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,5 **I** p3 jest 4,5 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,0 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,4 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,5 **I** p3 jest 4,9 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,2 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,5 **I** p1 jest 5,9 **TO** irys jest Versicolor
- JEŻELI** p3 jest 4,7 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,1 **TO** irys jest Versicolor
- JEŻELI** p3 jest 4,8 **I** p1 jest 5,9 **TO** irys jest Versicolor
- JEŻELI** p3 jest 4,6 **TO** irys jest Versicolor
- JEŻELI** p4 jest 1,6 **I** p1 jest 6,0 **TO** irys jest Versicolor
- JEŻELI** p4 jest 2,1 **TO** irys jest Virginica
- JEŻELI** p4 jest 2,3 **TO** irys jest Virginica
- JEŻELI** p3 jest 5,5 **TO** irys jest Virginica
- JEŻELI** p4 jest 2,0 **TO** irys jest Virginica

JEŻELI p_1 jest 7,3 **TO** irys jest Virginica
JEŻELI p_3 jest 6,0 **TO** irys jest Virginica
JEŻELI p_3 jest 5,1 **TO** irys jest Virginica
JEŻELI p_3 jest 5,8 **TO** irys jest Virginica
JEŻELI p_3 jest 6,1 **TO** irys jest Virginica
JEŻELI p_4 jest 2,4 **TO** irys jest Virginica
JEŻELI p_4 jest 1,8 **I** p_1 jest 6,2 **TO** irys jest Virginica
JEŻELI p_4 jest 1,7 **TO** irys jest Virginica
JEŻELI p_3 jest 6,7 **TO** irys jest Virginica
JEŻELI p_3 jest 5,0 **TO** irys jest Virginica
JEŻELI p_3 jest 5,7 **TO** irys jest Virginica
JEŻELI p_3 jest 4,9 **I** p_2 jest 2,7 **TO** irys jest Virginica

W kolejnym kroku dane zawarte w ciągu uczącym poddane zostały kwantyzacji tak, aby odpowiadająca im tablica decyzyjna była nadal deterministyczna. Program LERS ustalił dla poszczególnych atrybutów warunkowych następujące przedziały:

Tabela 3.10. Wynik kwantyzacji (klasyfikacja kwiatów irysa)

Atrybut	Zakres	Liczba próbek
p_1	(4,4; 5,05)	26
	(5,05; 5,65)	26
	(5,65; 6,15)	22
	(6,15; 6,65)	22
	(6,65; 7,9)	24
p_2	(2; 2,75)	24
	(2,75; 2,95)	18
	(2,95; 3,05)	21
	(3,05; 3,25)	21
	(3,25; 3,45)	17
	(3,45; 4,4)	19
p_3	(1; 2,6)	40
	(2,6; 4,85)	40
	(4,85; 6,9)	40
p_4	(0,1; 0,8)	40
	(0,8; 1,65)	42
	(1,65; 2,5)	38
Irys	Setosa	40
	Virginica	40
	Versicolor	40

Pierwotny plik wejściowy zastąpiono plikiem zamieszczonym poniżej. Każda wartość atrybutu decyzyjnego została zastąpiona identyfikatorem przedziału, do którego należy.

```

! Decision table produced by LERS(C version 1.0)
! First the attribute names list ...
!
[ p1 p2 p3 p4 irys ]
!
! Now comes the actual data. Please note that one example
! does NOT necessarily occupy one physical line
!
4.4..5.05 2.75..2.95 1..2.6 0.1..0.8 Setosa
4.4..5.05 2.95..3.05 1..2.6 0.1..0.8 Setosa
5.05..5.65 3.25..3.45 1..2.6 0.1..0.8 Setosa
...

```

Na podstawie tak przygotowanego pliku program LERS wygenerował 11 reguł zawierających razem 41 warunków:

JEŻELI p_3 jest $(1; 2,6)$ **TO** $irys$ jest Setosa

JEŻELI p_4 jest $(0,8; 1,65)$ **I** p_3 jest $(2,6; 4,85)$ **TO** $irys$ jest Versicolor

JEŻELI p_2 jest $(3,05; 3,25)$ **I** p_1 jest $(5,65; 6,15)$ **TO** $irys$ jest Versicolor

JEŻELI p_4 jest $(0,8; 1,65)$ **I** p_2 jest $(3,05; 3,25)$ **TO** $irys$ jest Versicolor

JEŻELI p_1 jest $(6,15; 6,65)$ **I** p_2 jest $(2; 2,75)$ **I** p_4 jest $(0,8; 1,65)$ **TO** $irys$ jest Versicolor

JEŻELI p_3 jest $(4,85; 6,9)$ **I** p_4 jest $(1,65; 2,5)$ **TO** $irys$ jest Virginica

JEŻELI p_3 jest $(4,85; 6,9)$ **I** p_2 jest $(2,75; 2,95)$ **TO** $irys$ jest Virginica

JEŻELI p_1 jest $(5,65; 6,15)$ **I** p_3 jest $(4,85; 6,9)$ **TO** $irys$ jest Virginica

JEŻELI p_4 jest $(1,65; 2,5)$ **I** p_2 jest $(2,75; 2,95)$ **TO** $irys$ jest Virginica

JEŻELI p_2 jest $(2,95; 3,05)$ **I** p_1 jest $(6,65; 7,9)$ **TO** $irys$ jest Virginica

JEŻELI p_2 jest $(2; 2,75)$ **I** p_4 jest $(1,65; 2,5)$ **TO** $irys$ jest Virginica

Reguły uzyskane w pierwszym i drugim podejściu zostały użyte do klasyfikacji próbek zawartych w zbiorze testowym. Wyniki obu eksperymentów przedstawiono w tabeli 3.11. Pierwsze cztery kolumny zawierają wartości atrybutów warunkowych dla próbek testowych, piąta kolumna to wynik wzorcowy (atrybut decyzyjny), kolumna szósta to wynik uzyskany z wykorzystaniem pierwszego zbioru reguł, a kolumna siódma to wynik uzyskany z wykorzystaniem drugiego zbioru reguł.

Analizując tabelę 3.11, można zauważyć m.in., że wstępna kwantyzacja danych, która dała w rezultacie zbiór reguł operujących na przedziałach (interwałach), prowadzi do uzyskania skuteczniejszego systemu wnioskującego.

3.7. Uwagi

Twórcą teorii zbiorów przybliżonych jest profesor Zdzisław Pawlak [161–164]. Podane w tym rozdziale definicje, jak również rozmaite zastosowania zbiorów przybliżonych przedstawia monografia [140], która jest pierwszym obszerniejszym opracowaniem na ten temat w języku polskim. Czytelnika zainteresowanego różnymi aspektami zbiorów

Tabela 3.11. Wyniki klasifikacji kwiatów irysa

p1	p2	p3	p4	Wzorzec	Klasyfikacja 1	Klasyfikacja 2
5,0	3,6	1,4	0,2	Setosa	Setosa	Setosa
4,9	3,1	1,5	0,1	Setosa	Setosa	Setosa
4,3	3,0	1,1	0,1	Setosa	Setosa	Setosa
5,0	3,0	1,6	0,2	Setosa	Setosa	Setosa
5,5	4,2	1,4	0,2	Setosa	Setosa	Setosa
5,1	3,4	1,5	0,2	Setosa	Setosa	Setosa
5,1	3,8	1,5	0,3	Setosa	Setosa	Setosa
5,1	3,5	1,4	0,3	Setosa	Setosa	Setosa
4,6	3,1	1,5	0,2	Setosa	Setosa	Setosa
5,1	3,8	1,9	0,4	Setosa	Setosa	Setosa
5,1	2,5	3,0	1,1	Versicolor	Versicolor	Versicolor
6,1	2,8	4,7	1,2	Versicolor	Versicolor	Versicolor
6,0	2,7	5,1	1,6	Versicolor	???	Virginica
5,5	2,4	3,8	1,1	Versicolor	Versicolor	Versicolor
4,9	2,4	3,3	1,0	Versicolor	Versicolor	Versicolor
6,7	3,0	5,0	1,7	Versicolor	Virginica	Virginica
6,2	2,2	4,5	1,5	Versicolor	Versicolor	Versicolor
6,8	2,8	4,8	1,4	Versicolor	Versicolor	Versicolor
5,7	2,8	4,5	1,3	Versicolor	Versicolor	Versicolor
5,8	2,6	4,0	1,2	Versicolor	Versicolor	Versicolor
6,3	2,5	5,0	1,9	Virginica	Virginica	Virginica
6,1	3,0	4,9	1,8	Virginica	???	Virginica
6,3	2,9	5,6	1,8	Virginica	???	Virginica
6,7	3,1	5,6	2,4	Virginica	Virginica	Virginica
5,8	2,8	5,1	2,4	Virginica	Virginica	Virginica
6,1	2,6	5,6	1,4	Virginica	Versicolor	Virginica
6,4	2,7	5,3	1,9	Virginica	???	Virginica
6,9	3,1	5,4	2,1	Virginica	Virginica	Virginica
6,0	3,0	4,8	1,8	Virginica	???	???
6,4	2,8	5,6	2,2	Virginica	???	Virginica

przybliżonych odsyłamy do bogatej literatury [66, 67, 158, 177, 180, 233]. W podręczniku 3.6 zastosowano program LERS do generacji reguł metodą zbiorów przybliżonych. Program ten został udostępniony na potrzeby tej książki dzięki uprzejmości profesora Jerzego Grzymały-Busse z Kansas University w USA.

Metody reprezentacji wiedzy z wykorzystaniem zbiorów rozmytych typu 1

4.1. Wprowadzenie

W życiu codziennym często spotykamy się ze zjawiskami i pojęciami, które mają charakter wieloznaczny i nieprecyzyjny. Posługując się klasyczną teorią zbiorów i logiką dwuwartościową, nie jesteśmy w stanie formalnie opisać takich zjawisk i pojęć. Z pomocą przychodzi nam teoria zbiorów rozmytych, która w ciągu ostatnich kilkunastu lat doczekała się wielu interesujących zastosowań.

W niniejszym rozdziale przedstawimy w sposób przystępny dla czytelnika podstawowe pojęcia i definicje teorii zbiorów rozmytych (podrozdz. 4.2–4.7). Następnie omówimy zagadnienia przybliżonego wnioskowania, tzn. wnioskowania na podstawie rozmytych przesłanek (podrozdz. 4.8). Kolejny punkt dotyczy problematyki konstrukcji rozmytych systemów wnioskujących (podrozdz. 4.9). Rozdział kończą przykłady zastosowań zbiorów rozmytych w zagadnieniach prognozowania, planowania oraz podejmowania decyzji.

4.2. Podstawowe pojęcia i definicje teorii zbiorów rozmytych

Z pomocą zbiorów rozmytych możemy formalnie określić pojęcia nieprecyzyjne i wieloznaczne, takie jak „wysoka temperatura”, „młody człowiek”, „średni wzrost” lub „duże miasto”. Przed podaniem definicji zbioru rozmytego musimy ustalić tzw. *obszar rozważań* (ang. *the universe of discourse*). W przypadku pojęcia wieloznacznego „duże pieniądze” inna suma będzie uważana za dużą, jeżeli ograniczymy się do obszaru rozważań $[0; 1000 \text{ zł}]$, a inna — jeżeli przyjmiemy przedział $[0; 1\,000\,000 \text{ zł}]$. Obszar rozważań, nazywany w dalszym ciągu przestrzenią lub zbiorem, będziemy najczęściej oznaczać literą **X**. Pamiętajmy, że **X** jest zbiorem nierożmytym.

DEFINICJA 4.1

Zbiorem rozmytym A w pewnej (niepustej) przestrzeni X, co zapisujemy jako $A \subseteq X$, nazywamy zbiór par

$$A = \{(x, \mu_A(x)); x \in X\}, \quad (4.1)$$

w którym

$$\mu_A : X \rightarrow [0, 1] \quad (4.2)$$

jest funkcją przynależności zbioru rozmytego A. Funkcja ta każdemu elementowi $x \in X$

przypisuje jego stopień przynależności do zbioru rozmytego A , przy czym można wyróżnić 3 przypadki:

- 1) $\mu_A(x) = 1$ oznacza pełną przynależność elementu x do zbioru rozmytego A , tzn. $x \in A$,
- 2) $\mu_A(x) = 0$ oznacza brak przynależności elementu x do zbioru rozmytego A , tzn. $x \notin A$,
- 3) $0 < \mu_A(x) < 1$ oznacza częściową przynależność elementu x do zbioru rozmytego A .

W literaturze stosuje się symboliczne zapisy zbiorów rozmytych. Jeżeli \mathbf{X} jest przestrzenią o skończonej liczbie elementów, $\mathbf{X} = \{x_1, \dots, x_n\}$, to zbiór rozmyty $A \subseteq \mathbf{X}$ zapisuje się jako

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots + \frac{\mu_A(x_n)}{x_n} = \sum_{i=1}^n \frac{\mu_A(x_i)}{x_i}. \quad (4.3)$$

Warto wspomnieć, że elementami $x_i \in \mathbf{X}$ mogą być nie tylko liczby, ale również osoby, przedmioty lub inne pojęcia. Zapis (4.3) ma charakter symboliczny. Kreska ułamkowa nie oznacza dzielenia, ale oznacza przyporządkowanie poszczególnym elementom x_1, \dots, x_n stopni przynależności $\mu_A(x_1), \dots, \mu_A(x_n)$. Innymi słowy, zapis

$$\frac{\mu_A(x_i)}{x_i} \quad i = 1, \dots, n \quad (4.4)$$

oznacza parę

$$(x_i, \mu_A(x_i)) \quad i = 1, \dots, n. \quad (4.5)$$

Podobnie znak „+” nie oznacza dodawania, ale sumę mnogościową elementów (4.5). Warto zauważyć, że w podobnej konwencji można symbolicznie zapisywać zbiory nie-rozmyte. Na przykład zbiór ocen w szkole symbolicznie zapisujemy jako

$$\mathbf{D} = 1 + 2 + 3 + 4 + 5 + 6, \quad (4.6)$$

co jest równoznaczne z zapisem

$$\mathbf{D} = \{1, 2, 3, 4, 5, 6\}. \quad (4.7)$$

Jeżeli \mathbf{X} jest przestrzenią o nieskończonej liczbie elementów, to zbiór rozmyty $A \subseteq \mathbf{X}$ symbolicznie zapisujemy jako

$$A = \int_{\mathbf{X}} \frac{\mu_A(x)}{x}. \quad (4.8)$$

Przykład 4.1

Załóżmy, że $\mathbf{X} = \mathbf{N}$ jest zbiorem liczb naturalnych. Określmy pojęcie zbioru liczb naturalnych „bliskich liczby 7”. Można tego dokonać, definiując następujący zbiór rozmyty $A \subseteq \mathbf{X}$:

$$A = \frac{0,2}{4} + \frac{0,5}{5} + \frac{0,8}{6} + \frac{1}{7} + \frac{0,8}{8} + \frac{0,5}{9} + \frac{0,2}{10}. \quad (4.9)$$

Przykład 4.2

Jeżeli $X = \mathbf{R}$, gdzie \mathbf{R} jest zbiorem liczb rzeczywistych, to zbiór liczb rzeczywistych „bliskich liczby 7” definiujemy poprzez funkcję przynależności postaci, np.

$$\mu_A(x) = \frac{1}{1 + (x - 7)^2}. \quad (4.10)$$

Zatem zbiór rozmyty liczb rzeczywistych „bliskich liczby 7” zapisujemy jako

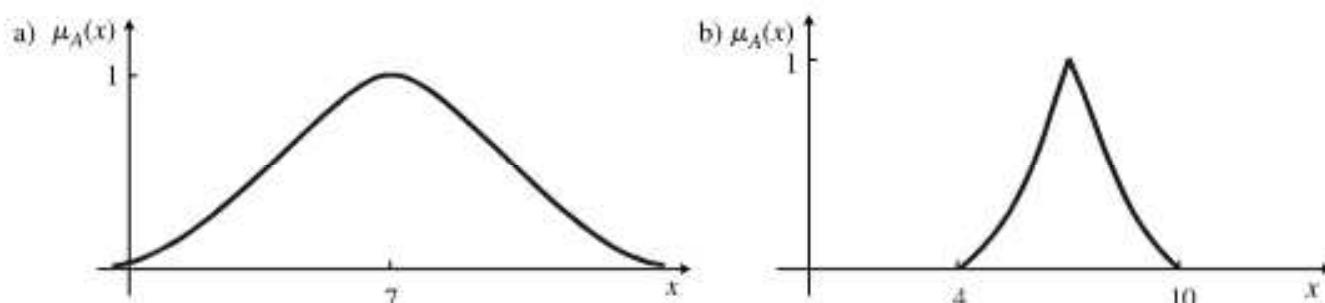
$$A = \int_X \frac{[1 + (x - 7)^2]^{-1}}{x}. \quad (4.11)$$

Uwaga 4.1

Zbiory rozmyte liczb naturalnych oraz rzeczywistych „bliskich liczby 7” można zapisać na wiele sposobów. Na przykład funkcję przynależności (4.10) można zastąpić wzorem

$$\mu_A(x) = \begin{cases} 1 - \sqrt{\frac{|x - 7|}{3}}, & \text{jeżeli } 4 \leq x \leq 10, \\ 0, & \text{w przeciwnym razie.} \end{cases} \quad (4.12)$$

Na rysunkach 4.1a oraz 4.1b przedstawiono dwie funkcje przynależności zbioru rozmytego A liczb rzeczywistych „bliskich liczby 7”.



Rys. 4.1. Ilustracja do przykładu 4.2: funkcje przynależności zbioru rozmytego liczb rzeczywistych „bliskich liczby 7”

Przykład 4.3

Sformalizujmy określenie nieprecyzyjne „odpowiednia do kąpieli temperatura wody w Bałtyku”. Ustalmy obszar rozważań jako zbiór $X = [15^\circ, \dots, 25^\circ]$. Wczasowicz I, najlepiej czujący się w temperaturze 21° , zdefiniowałby następujący zbiór rozmyty:

$$A = \frac{0,1}{16} + \frac{0,3}{17} + \frac{0,5}{18} + \frac{0,8}{19} + \frac{0,95}{20} + \frac{1}{21} + \frac{0,9}{22} + \frac{0,8}{23} + \frac{0,75}{24} + \frac{0,7}{25}. \quad (4.13)$$

Wczasowicz II, preferujący temperaturę 20° , podałby inną definicję tego zbioru

$$B = \frac{0,1}{15} + \frac{0,2}{16} + \frac{0,4}{17} + \frac{0,7}{18} + \frac{0,9}{19} + \frac{1}{20} + \frac{0,9}{21} + \frac{0,85}{22} + \frac{0,8}{23} + \frac{0,75}{24} + \frac{0,7}{25}. \quad (4.14)$$

Z pomocą zbiorów rozmytych A i B sformalizowaliśmy nieprecyzyjne określenie „odpowiednia do kąpieli temperatura wody w Bałtyku”.

Uwaga 4.2

Należy podkreślić, że teoria zbiorów rozmytych opisuje niepewność w innym sensie aniżeli rachunek prawdopodobieństwa. Za pomocą rachunku prawdopodobieństwa możemy wyznaczyć na przykład prawdopodobieństwo wyrzucenia 4, 5 lub 6 podczas rzutu kostką. Oczywiście, prawdopodobieństwo to wynosi 0,5. Natomiast za pomocą zbiorów rozmytych możemy opisać nieprecyzyjne stwierdzenie „wyrzucenie dużej liczby oczek”. Odpowiedni zbiór rozmyty może mieć postać

$$A = \frac{0,6}{4} + \frac{0,8}{5} + \frac{1}{6}$$

lub

$$A = \frac{0,1}{3} + \frac{0,5}{4} + \frac{0,85}{5} + \frac{1}{6}.$$

Jednym podobieństwem pomiędzy teorią zbiorów rozmytych a teorią rachunku prawdopodobieństwa jest fakt, że zarówno funkcja przynależności zbioru rozmytego, jak i prawdopodobieństwo przyjmują wartości w zbiorze [0, 1].

W niektórych zastosowaniach używa się standardowych postaci funkcji przynależności. Poniżej wyszczególnimy te funkcje oraz przedstawimy ich interpretacje graficzne.

1. Funkcję *singleton* definiujemy następująco:

$$\mu_A(x) = \begin{cases} 1, & \text{jeżeli } x = \bar{x}, \\ 0, & \text{jeżeli } x \neq \bar{x}. \end{cases} \quad (4.15)$$

Singleton jest specyficzną funkcją przynależności, gdyż przyjmuje wartość 1 tylko w jednym punkcie przestrzeni rozważeń, należącym w pełni do zbioru rozmytego. W pozostałych punktach przyjmuje wartość 0. Taka funkcja przynależności charakteryzuje jednoelementowy zbiór rozmyty. Jedynym elementem w pełni należącym do zbioru rozmytego A jest punkt \bar{x} . Funkcja przynależności typu singleton wykorzystywana jest głównie do realizacji operacji rozmywania stosowanej w rozmytych systemach wnioskujących.

2. *Gaussowska* funkcja przynależności (rys. 4.2) jest opisana wzorem

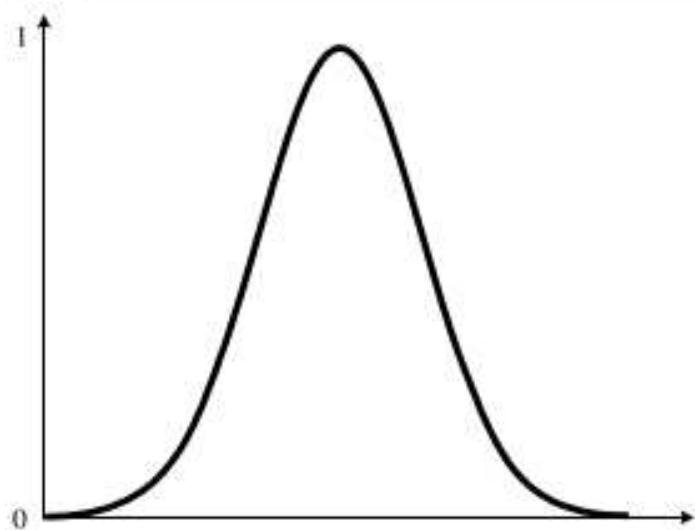
$$\mu_A(x) = \exp \left(- \left(\frac{x - \bar{x}}{\sigma} \right)^2 \right), \quad (4.16)$$

w którym \bar{x} jest środkiem, a σ określa szerokość krzywej gaussowskiej. Jest to najczęściej spotykana funkcja przynależności.

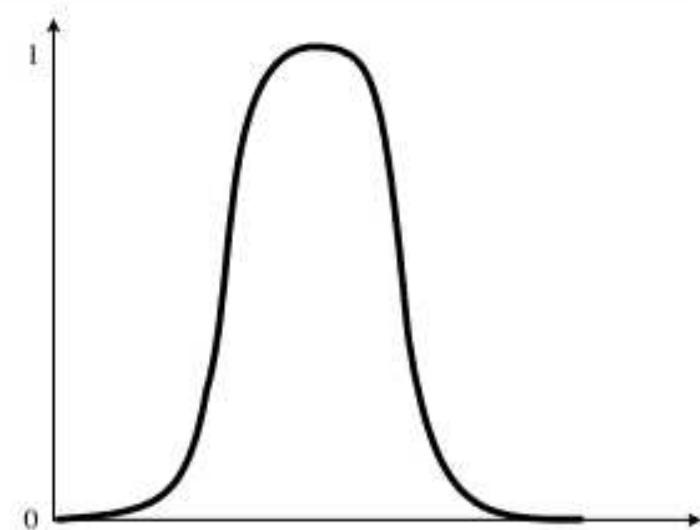
3. Funkcja przynależności typu *dzwonowego* (rys. 4.3) jest postaci

$$\mu(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}, \quad (4.17)$$

gdzie parametr a określa jej szerokość, parametr b nachylenie, natomiast parametr c środek.



Rys. 4.2. Wykres gaussowskiej funkcji przynależności

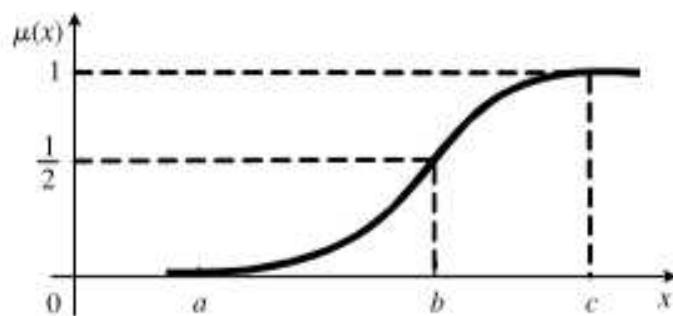


Rys. 4.3. Wykres funkcji przynależności typu dzwonowego

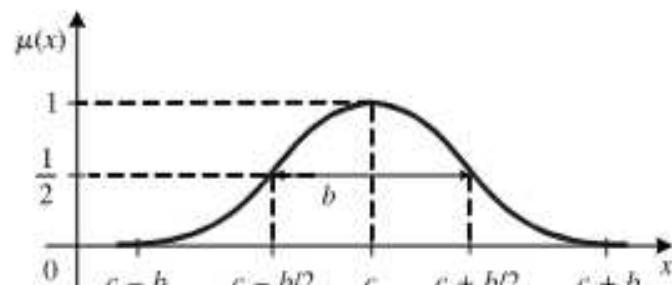
4. Funkcja przynależności *klasy s* (rys. 4.4) jest zdefiniowana jako

$$s(x; a, b, c) = \begin{cases} 0 & \text{dla } x \leq a, \\ 2 \left(\frac{x-a}{c-a} \right)^2 & \text{dla } a < x \leq b, \\ 1 - 2 \left(\frac{x-c}{c-a} \right)^2 & \text{dla } b < x \leq c, \\ 1 & \text{dla } x > c. \end{cases} \quad (4.18)$$

gdzie $b = (a+c)/2$. Wykres funkcji przynależności należącej do tej klasy ma postać graficzną przypominającą literę „s”, przy czym jej kształt zależy od doboru parametrów a , b i c . W punkcie $x = b = (a+c)/2$ funkcja przynależności klasy *s* przyjmuje wartość 0,5.



Rys. 4.4. Funkcja przynależności klasy *s*



Rys. 4.5. Funkcja przynależności klasy *π*

5. Funkcja przynależności *klasy π* (rys. 4.5) jest zdefiniowana poprzez funkcję przynależności klasy *s*

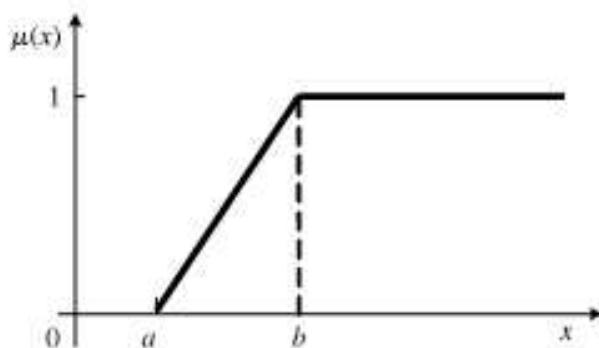
$$\pi(x; b, c) = \begin{cases} s(x; c-b, c-b/2, c) & \text{dla } x \leq c, \\ 1 - s(x; c, c+b/2, c+b) & \text{dla } x > c. \end{cases} \quad (4.19)$$

Funkcja przynależności klasy *π* przyjmuje wartości zerowe dla $x \geq c+b$ oraz $x \leq c-b$. W punktach $x = c \pm b/2$ jej wartość wynosi 0,5.

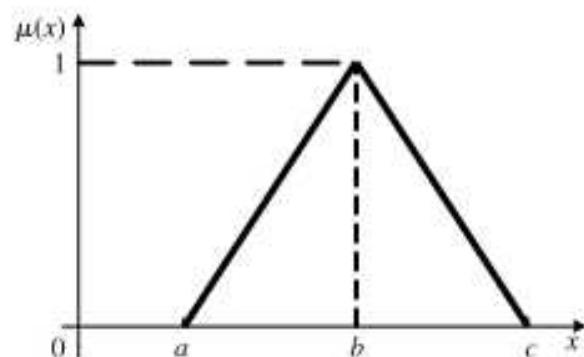
6. Funkcja przynależności klasy γ (rys. 4.6) jest dana wzorem

$$\gamma(x; a, b) = \begin{cases} 0 & \text{dla } x \leq a, \\ \frac{x-a}{b-a} & \text{dla } a < x \leq b, \\ 1 & \text{dla } x > b. \end{cases} \quad (4.20)$$

Czytelnik z łatwością zauważa analogie między kształtami funkcji przynależności klasy s i γ .



Rys. 4.6. Funkcja przynależności klasy γ



Rys. 4.7. Funkcja przynależności klasy t

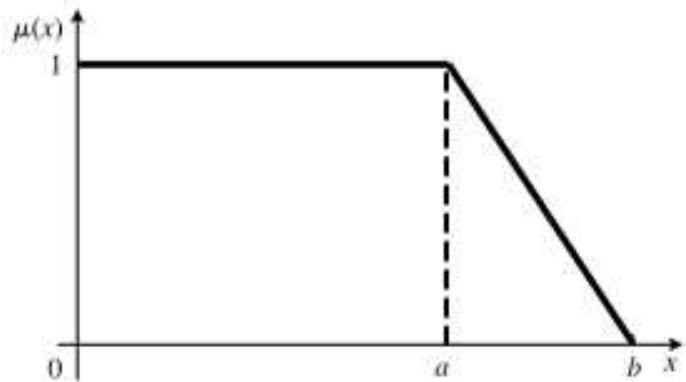
7. Funkcja przynależności klasy t (rys. 4.7) jest zdefiniowana następująco:

$$t(x; a, b, c) = \begin{cases} 0 & \text{dla } x \leq a, \\ \frac{x-a}{b-a} & \text{dla } a < x \leq b, \\ \frac{c-x}{c-b} & \text{dla } b < x \leq c, \\ 0 & \text{dla } x > c. \end{cases} \quad (4.21)$$

W niektórych zastosowaniach funkcja przynależności klasy t może być alternatywna w stosunku do funkcji klasy π .

8. Funkcja przynależności klasy L (rys. 4.8) jest określona wzorem

$$L(x; a, b) = \begin{cases} 1 & \text{dla } x \leq a, \\ \frac{b-x}{b-a} & \text{dla } a < x \leq b, \\ 0 & \text{dla } x > b. \end{cases} \quad (4.22)$$



Rys. 4.8. Funkcja przynależności klasy L

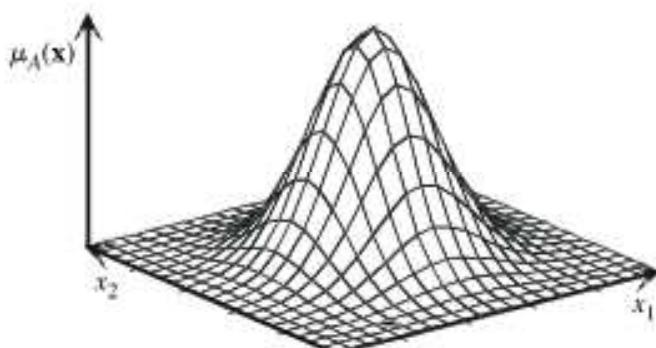
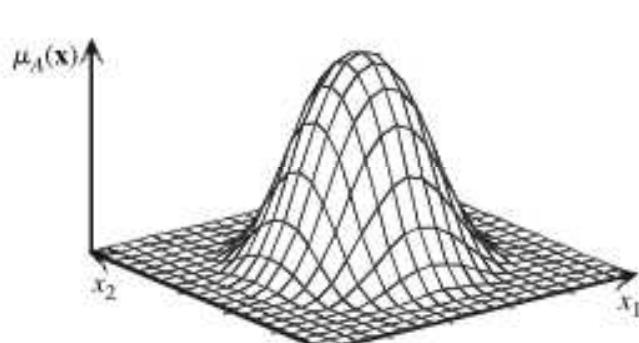
Uwaga 4.3

Powyżej podaliśmy przykłady standardowych funkcji przynależności dla zbiorów rozmytych określonych w przestrzeni liczb rzeczywistych, tzn. $\mathbf{X} \subset \mathbf{R}$. Gdy $\mathbf{X} \subset \mathbf{R}^n$, $\mathbf{x} = [x_1, \dots, x_n]^T$, $n > 1$, możemy rozróżnić dwa przypadki. Pierwszy zachodzi, gdy założymy niezależność poszczególnych zmiennych x_i , $i = 1, \dots, n$. Wówczas wielowymiarowe funkcje przynależności tworzymy, stosując definicję iloczynu kartezjańskiego zbiorów rozmytych (definicja 4.14) oraz korzystając ze standardowych funkcji przynależności jednej zmiennej. W przypadku gdy zmienne x_i są zależne, stosujemy wielowymiarowe funkcje przynależności. Poniżej podano trzy przykłady takich funkcji.

1. Funkcja przynależności *klasy* Π (rys. 4.9) jest zdefiniowana następująco:

$$\mu_A(\mathbf{x}) = \begin{cases} 1 - 2 \cdot \left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\alpha} \right)^2 & \text{dla } \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \frac{1}{2}\alpha, \\ 2 \cdot \left(1 - \frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\alpha} \right)^2 & \text{dla } \frac{1}{2}\alpha < \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \alpha, \\ 0 & \text{dla } \|\mathbf{x} - \bar{\mathbf{x}}\| > \alpha, \end{cases} \quad (4.23)$$

gdzie $\bar{\mathbf{x}}$ jest środkiem funkcji przynależności, a $\alpha > 0$ parametrem określającym jej rozpiętość.



Rys. 4.9. Dwuwymiarowa funkcja przynależności klasy Π

Rys. 4.10. Radialna funkcja przynależności klasy Π

2. *Radialna* funkcja przynależności (rys. 4.10) jest postaci

$$\mu_A(\mathbf{x}) = e^{\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|^2}{2\sigma^2}}, \quad (4.24)$$

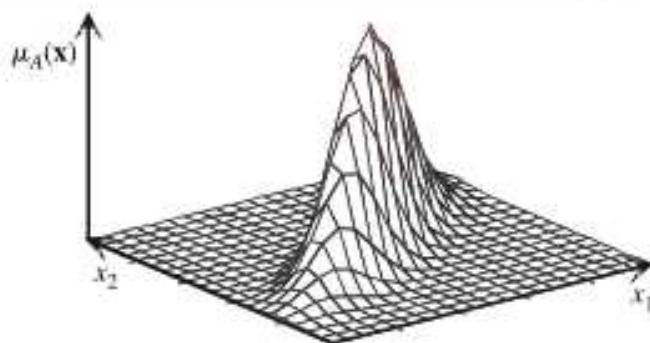
gdzie $\bar{\mathbf{x}}$ jest środkiem, natomiast wartość parametru σ wpływa na kształt tej funkcji.

3. *Elipsoidalna* funkcja przynależności (rys. 4.11) jest zdefiniowana następująco:

$$\mu_A(\mathbf{x}) = \exp \left(-\frac{(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{Q}^{-1} (\mathbf{x} - \bar{\mathbf{x}})}{\alpha} \right), \quad (4.25)$$

gdzie $\bar{\mathbf{x}}$ jest środkiem, $\alpha > 0$ jest parametrem określającym rozpiętość tej funkcji, a \mathbf{Q} jest tzw. macierzą kowariancji. Modyfikując tę macierz, można modelować kształt tej funkcji.

Podamy teraz dwa przykłady ilustrujące zastosowania standardowych funkcji przynależności jednej zmiennej.

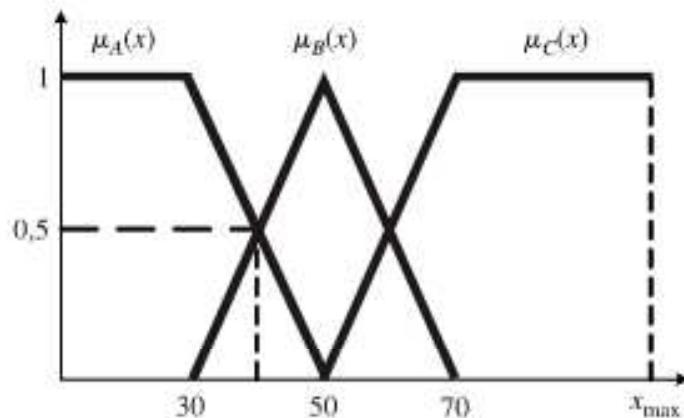


Rys. 4.11. Elipsoidalna funkcja przynależności

Przykład 4.4

Rozważmy trzy nieprecyzyjne stwierdzenia:

- 1) „mała szybkość samochodu”,
- 2) „średnia szybkość samochodu”,
- 3) „duża szybkość samochodu”.



Rys. 4.12. Ilustracja do przykładu 4.4: funkcje przynależności zbiorów rozmytych „mała” ($\mu_A(x)$), „średnia” ($\mu_B(x)$), „duża” ($\mu_C(x)$) szybkość samochodu

Jako obszar rozważań \mathbf{X} przyjmiemy przedział $[0, x_{\max}]$, gdzie x_{\max} jest prędkością maksymalną. Na rysunku 4.12 przedstawiono zbiory rozmyte A , B , C odpowiadające powyższym stwierdzeniom. Zauważmy, że funkcja przynależności zbioru A jest klasy L , zbioru B jest klasy t , natomiast zbioru C jest klasy γ . W ustalonym punkcie $x = 40$ km/h funkcja przynależności zbioru rozmytego „mała szybkość samochodu” przyjmuje wartość 0,5, tzn. $\mu_A(40) = 0.5$. Taką samą wartość przyjmuje funkcja przynależności zbioru rozmytego „średnia szybkość samochodu”, tzn. $\mu_B(40) = 0.5$, natomiast $\mu_C(40) = 0$.

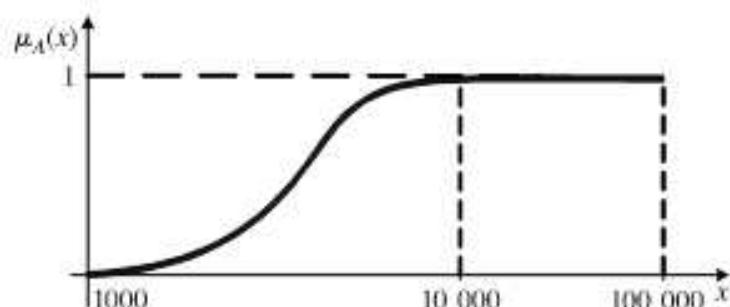
Przykład 4.5

Na rysunku 4.13 pokazano funkcję przynależności zbioru rozmytego „duże pieniądze”. Jest to funkcja klasy s , przy czym $\mathbf{X} = [0, 100\ 000 \text{ zł}]$, $a = 1000 \text{ zł}$, $c = 10\ 000 \text{ zł}$. Tak więc na pewno sumy większe od 10 000 zł możemy uważać jako „duże”, gdyż wówczas wartości funkcji przynależności są równe 1. Kwoty mniejsze od 1000 zł nie są „duże”, gdyż odpowiadające im wartości funkcji przynależności są równe 0. Taka definicja zbioru rozmytego „duże pieniądze” ma charakter subiektywny. Czytelnik może mieć swój pogląd na temat wieloznacznego stwierdzenia „duże pieniądze”. Pogląd ten będą odzwierciedlać inne wartości parametrów a i c funkcji klasy s .

DEFINICJA 4.2

Zbiór elementów przestrzeni \mathbf{X} , dla których $\mu_A(x) > 0$, nazywamy *nośnikiem zbioru rozmytego A* i oznaczamy $\text{supp } A$ (ang. *support*). Zapisujemy

$$\text{supp } A = \{x \in \mathbf{X}; \mu_A(x) > 0\}. \quad (4.26)$$



Rys. 4.13. Ilustracja do przykładu 4.5: funkcja przynależności zbioru rozmytego „dużo pieniędzy”

DEFINICJA 4.3

Wysokość zbioru rozmytego A oznaczamy $h(A)$ i określamy jako

$$h(A) = \sup_{x \in \mathbf{X}} \mu_A(x). \quad (4.27)$$

Przykład 4.6

Jeżeli $\mathbf{X} = \{1, 2, 3, 4, 5\}$ oraz

$$A = \frac{0,2}{1} + \frac{0,4}{2} + \frac{0,7}{4}, \quad (4.28)$$

to $\text{supp } A = \{1, 2, 4\}$.

Jeżeli $\mathbf{X} = \{1, 2, 3, 4\}$ oraz

$$A = \frac{0,3}{2} + \frac{0,8}{3} + \frac{0,5}{4}, \quad (4.29)$$

to $h(A) = 0,8$.

DEFINICJA 4.4

Zbiór rozmyty A nazywamy *normalnym* wtedy i tylko wtedy, gdy $h(A) = 1$. Jeżeli zbiór rozmyty A nie jest normalny, to można go znormalizować za pomocą przekształcenia

$$\mu_{A_{zn}}(x) = \frac{\mu_A(x)}{h(A)}, \quad (4.30)$$

gdzie $h(A)$ jest wysokością tego zbioru.

Przykład 4.7

Zbiór rozmyty

$$A = \frac{0,1}{2} + \frac{0,5}{4} + \frac{0,3}{6} \quad (4.31)$$

po znormalizowaniu przybiera postać

$$A_{zn} = \frac{0,2}{2} + \frac{1}{4} + \frac{0,6}{6}. \quad (4.32)$$

DEFINICJA 4.5

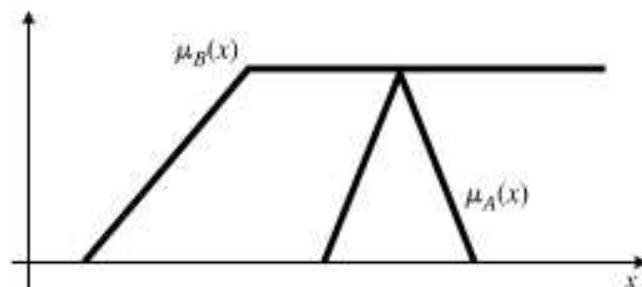
Zbiór rozmyty A jest *pusty*, co zapisujemy $A = \emptyset$, wtedy i tylko wtedy, gdy $\mu_A(x) = 0$ dla każdego $x \in \mathbf{X}$.

DEFINICJA 4.6

Zbiór rozmyty A *zawiera się* w zbiorze rozmytym B , co zapisujemy $A \subset B$, wtedy i tylko wtedy, gdy

$$\mu_A(x) \leq \mu_B(x) \quad (4.33)$$

dla każdego $x \in \mathbf{X}$. Przykład *inkluzji* (zawierania się) zbioru rozmytego A w zbiorze rozmytym B ilustruje rysunek 4.14.



Rys. 4.14. Inkluzja zbioru rozmytego A w zbiorze rozmytym B

DEFINICJA 4.7

Zbiór rozmyty A jest *równy* zbiorowi rozmytemu B , co zapisujemy $A = B$, wtedy i tylko wtedy, gdy

$$\mu_A(x) = \mu_B(x) \quad (4.34)$$

dla każdego $x \in \mathbf{X}$. Powyższa definicja, podobnie jak definicja 4.6, nie jest „elastyczna”, gdyż nie uwzględnia przypadku, gdy wartości funkcji przynależności $\mu_A(x)$ i $\mu_B(x)$ są prawie równe. Możemy wówczas wprowadzić pojęcie stopnia równości zbiorów rozmytych A i B jako np.

$$E(A = B) = 1 - \max_{x \in T} |\mu_A(x) - \mu_B(x)|, \quad (4.35)$$

gdzie $T = \{x \in \mathbf{X} : \mu_A(x) \neq \mu_B(x)\}$. Różne definicje stopnia inkluzyj oraz stopnia równości zbiorów rozmytych szczegółowo przedstawiono w monografii [94].

DEFINICJA 4.8

α -*Przekrojem* zbioru rozmytego $A \subseteq \mathbf{X}$, oznaczanym A_α , nazywamy następujący zbiór nierozmyty:

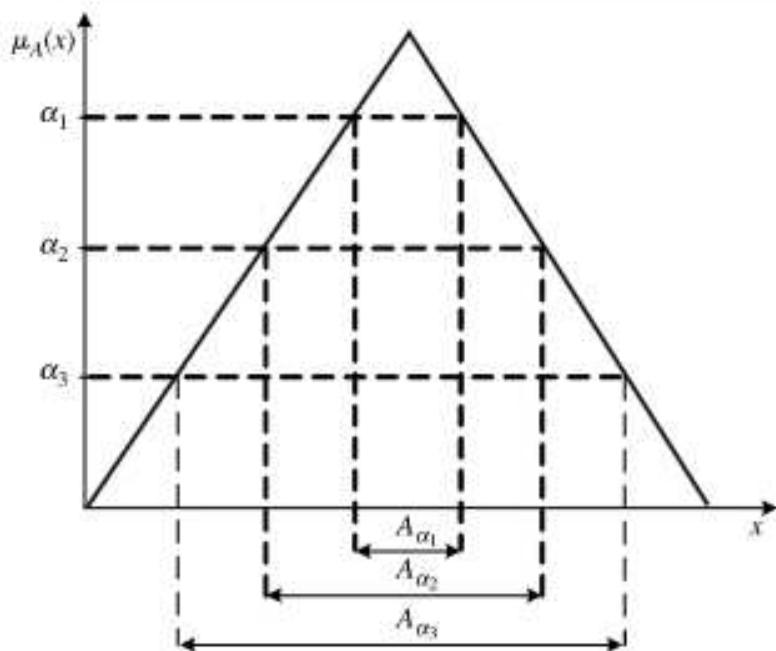
$$A_\alpha = \{x \in \mathbf{X} : \mu_A(x) \geq \alpha\}, \quad \forall \alpha \in [0, 1], \quad (4.36)$$

czyli zbiór określony przez funkcję charakterystyczną

$$\chi_{A_\alpha}(x) = \begin{cases} 1 & \text{dla } \mu_A(x) \geq \alpha, \\ 0 & \text{dla } \mu_A(x) < \alpha. \end{cases} \quad (4.37)$$

Definicję α -przekroju zbioru rozmytego ilustruje rysunek 4.15. Łatwo zauważyc, że zachodzi następująca implikacja:

$$\alpha_2 < \alpha_1 \Rightarrow A_{\alpha_1} \subset A_{\alpha_2}. \quad (4.38)$$



Rys. 4.15. Ilustracja definicji α -przekroju zbioru rozmytego A

Przykład 4.8

Rozważmy zbiór rozmyty $A \subseteq \mathbf{X}$

$$A = \frac{0.1}{2} + \frac{0.3}{4} + \frac{0.7}{5} + \frac{0.8}{8} + \frac{1}{10}, \quad (4.39)$$

przy czym $\mathbf{X} = \{1, \dots, 10\}$. Zgodnie z definicją 4.8 poszczególne α -przekroje określamy następująco:

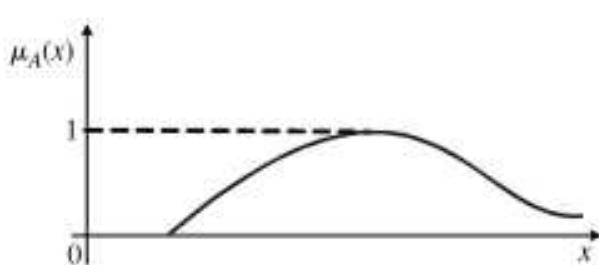
$$\begin{aligned} A_0 &= \mathbf{X} = \{1, \dots, 10\}, \\ A_{0,1} &= \{2, 4, 5, 8, 10\}, \\ A_{0,3} &= \{4, 5, 8, 10\}, \\ A_{0,7} &= \{5, 8, 10\}, \\ A_{0,8} &= \{8, 10\}, \\ A_1 &= \{10\}. \end{aligned}$$

DEFINICJA 4.9

Zbiór rozmyty $A \subseteq \mathbf{R}$ jest *wypukły* wtedy i tylko wtedy, gdy dla dowolnych $x_1, x_2 \in \mathbf{R}$ i $\lambda \in [0, 1]$ zachodzi

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min\{\mu_A(x_1), \mu_A(x_2)\}. \quad (4.40)$$

Na rysunku 4.16 przedstawiono przykład zbioru rozmytego wypukłego.



Rys. 4.16. Zbiór rozmyty wypukły



Rys. 4.17. Zbiór rozmyty wklęsły

DEFINICJA 4.10

Zbiór rozmyty $A \subseteq \mathbf{R}$ jest *wklęsły* wtedy i tylko wtedy, gdy istnieją takie punkty $x_1, x_2 \in \mathbf{R}$ i $\lambda \in [0, 1]$, że spełniona jest nierówność

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] < \min\{\mu_A(x_1), \mu_A(x_2)\}. \quad (4.41)$$

Rysunek 4.17 ilustruje zbiór rozmyty wklęsły.

4.3. Operacje na zbiorach rozmytych

W tym podrozdziale podamy podstawowe operacje na zbiorach rozmytych, zarówno operacje mnogościowe, jak i algebraiczne.

DEFINICJA 4.11

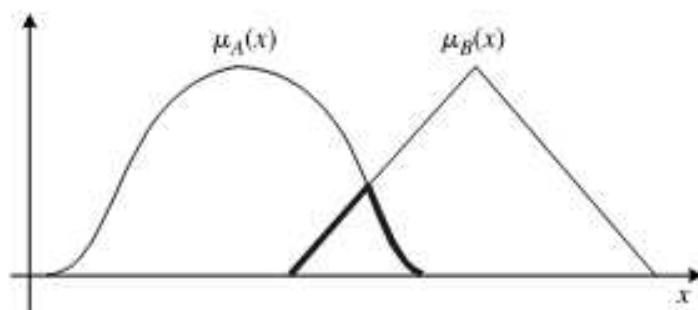
Przecięciem zbiorów rozmytych $A, B \subseteq \mathbf{X}$ jest zbiór rozmyty $A \cap B$ o funkcji przynależności

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad (4.42)$$

dla każdego $x \in \mathbf{X}$. Działanie tej operacji przedstawiono graficznie na rysunku 4.18. Przecięcie zbiorów rozmytych A_1, A_2, \dots, A_n określone jest funkcją przynależności

$$\mu_{A_1 \cap A_2 \cap \dots \cap A_n}(x) = \min[\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)] \quad (4.43)$$

dla każdego $x \in \mathbf{X}$.



Rys. 4.18. Działanie operacji przecięcia zbiorów rozmytych

Uwaga 4.4

W literaturze oprócz definicji *przecięcia* (ang. *intersection*) zbiorów rozmytych spotyka się również definicję *iloczynu algebraicznego* (ang. *algebraic product*) tych zbiorów. Iloczynem algebraicznym zbiorów rozmytych A i B jest zbiór rozmyty $C = A \cdot B$ zdefiniowany następująco:

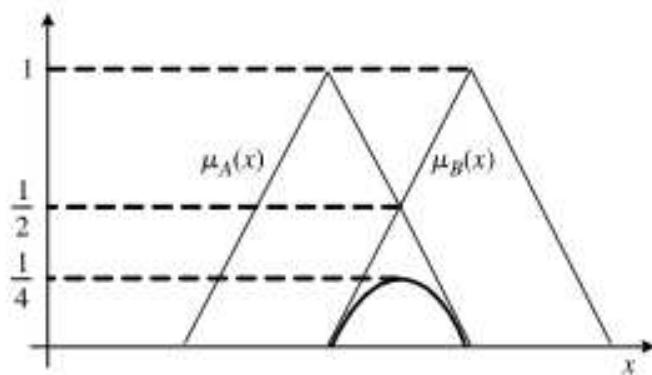
$$C = \{(x, \mu_A(x) \cdot \mu_B(x)) \mid x \in \mathbf{X}\}. \quad (4.44)$$

Działanie operacji iloczynu algebraicznego ilustruje rysunek 4.19.

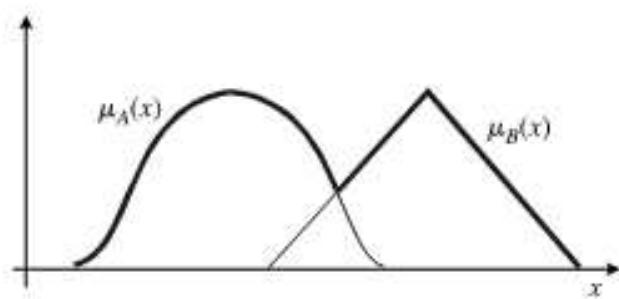
DEFINICJA 4.12

Sumą zbiorów rozmytych A i B jest zbiór rozmyty $A \cup B$ określony funkcją przynależności

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad (4.45)$$



Rys. 4.19. Działanie operacji iloczynu algebraicznego



Rys. 4.20. Działanie operacji sumy zbiorów rozmytych

dla każdego $x \in \mathbf{X}$. Działanie tej operacji ilustruje rysunek 4.20. Funkcja przynależności sumy zbiorów rozmytych A_1, A_2, \dots, A_n wyraża się zależnością

$$\mu_{A_1 \cup A_2 \cup \dots \cup A_n}(x) = \max[\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)] \quad (4.46)$$

dla każdego $x \in \mathbf{X}$.

Przykład 4.9

Założymy, że $\mathbf{X} = \{1, 2, 3, 4, 5, 6, 7\}$ oraz

$$A = \frac{0,9}{3} + \frac{1}{4} + \frac{0,6}{6}, \quad (4.47)$$

$$B = \frac{0,7}{3} + \frac{1}{5} + \frac{0,4}{6}. \quad (4.48)$$

Zgodnie z definicją 4.11 otrzymujemy

$$A \cap B = \frac{0,7}{3} + \frac{0,4}{6}. \quad (4.49)$$

Na mocy definicji 4.12 mamy

$$A \cup B = \frac{0,9}{3} + \frac{1}{4} + \frac{1}{5} + \frac{0,6}{6}. \quad (4.50)$$

Natomiast iloczyn zbiorów rozmytych A i B dany wzorem (4.44) przybiera postać

$$A \cdot B = \frac{0,63}{3} + \frac{0,24}{6}. \quad (4.51)$$

W literaturze znane jest tzw. *twierdzenie o dekompozycji*. Pozwala ono przedstawić dowolny zbiór rozmyty A w postaci sumy zbiorów rozmytych generowanych przez α -przekroje zbioru A .

TWIERDZENIE 4.1

Każdy zbiór rozmyty $A \subseteq \mathbf{X}$ można przedstawić w postaci

$$A = \bigcup_{\alpha \in [0,1]} \alpha A_\alpha, \quad (4.52)$$

gdzie αA_α oznacza zbiór rozmyty, którego elementom przypisano następujące stopnie przynależności:

$$\mu_{\alpha A_\alpha}(x) = \begin{cases} \alpha & \text{dla } x \in A_\alpha, \\ 0 & \text{dla } x \notin A_\alpha. \end{cases} \quad (4.53)$$

Przykład 4.10

Dokonamy dekompozycji zbioru rozmytego (4.39). Zgodnie ze wzorem (4.52) otrzymujemy

$$\begin{aligned} A &= \left(\frac{0,1}{2} + \frac{0,1}{4} + \frac{0,1}{5} + \frac{0,1}{8} + \frac{0,1}{10} \right) \cup \left(\frac{0,3}{4} + \frac{0,3}{5} + \frac{0,3}{8} + \frac{0,3}{10} \right) \\ &\quad \cup \left(\frac{0,7}{5} + \frac{0,7}{8} + \frac{0,7}{10} \right) \cup \left(\frac{0,8}{8} + \frac{0,8}{10} \right) \cup \frac{1}{10} \\ &= \frac{0,1}{2} + \frac{0,3}{4} + \frac{0,7}{5} + \frac{0,8}{8} + \frac{1}{10}. \end{aligned} \quad (4.54)$$

Uwaga 4.5

Definicje 4.11 i 4.12 nie są jedynymi znanymi w literaturze definicjami przecięcia oraz sumy zbiorów rozmytych. Zamiast powtórzonych poniżej równości (4.42) i (4.45)

$$\begin{cases} \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \\ \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \end{cases}$$

możemy spotkać alternatywne definicje wykorzystujące pojęcie tzw. *t-normy* i *t-konormy*. Tak więc operacja (4.42) jest przykładem działania *t-normy* (operacja przecięcia), natomiast operacja (4.45) jest przykładem działania *t-konormy* (operacja sumy). W podrozdziale 4.6 zostaną przedstawione formalne definicje *t-normy* i *t-konormy* oraz ogólniejsze definicje przecięcia i sumy zbiorów rozmytych.

Uwaga 4.6

W literaturze znane są próby analitycznego znalezienia „najlepszych” operacji przecięcia i sumy zbiorów rozmytych. Na przykład, Bellman i Giertz [8] postawili i rozwiązały problem znalezienia dwóch funkcji f i g

$$f, g : [0, 1] \times [0, 1] \rightarrow [0, 1]$$

takich, że

$$\mu_{A \cap B}(x) = f(\mu_A(x), \mu_B(x)), \quad (4.55)$$

$$\mu_{A \cup B}(x) = g(\mu_A(x), \mu_B(x)). \quad (4.56)$$

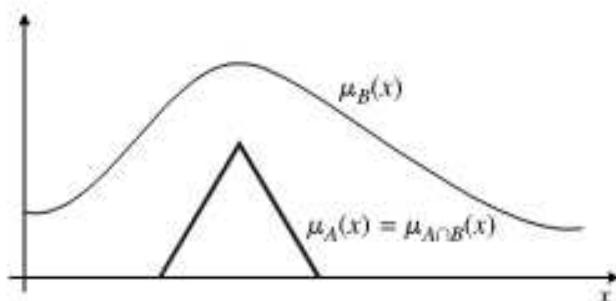
Autorzy wymienionej publikacji narzucili szereg warunków na funkcje f i g , po czym wykazali, że jedynie operacje (4.42) i (4.45) spełniają te warunki. Nie oznacza to, że operacje (4.42) i (4.45) są adekwatne we wszystkich zastosowaniach, np. jeżeli

$$\mu_A(x) < \mu_B(x), \quad \forall x \in X, \quad (4.57)$$

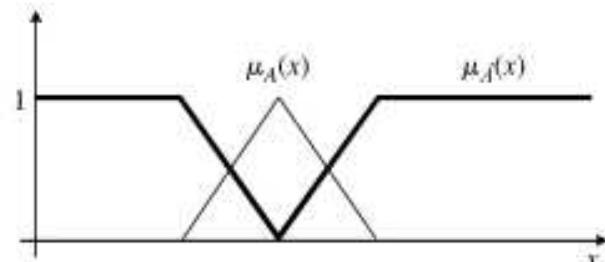
to w wyniku operacji (4.42) otrzymujemy

$$\mu_{A \cap B}(x) = \mu_A(x) \quad (4.58)$$

niezależnie od wielkości $\mu_B(x)$. Innymi słowy, funkcja przynależności zbioru rozmytego B nie ma żadnego wpływu na wyznaczenie przecięcia zbiorów rozmytych A i B . Fakt ten ilustruje rysunek 4.21. W takiej sytuacji bardziej sensowne wydaje się zastosowanie np. wzoru (4.44) jako operacji przecięcia. Wówczas przecięcie dwóch zbiorów rozmytych będzie tożsame z iloczynem tych zbiorów (patrz uwaga 4.4).



Rys. 4.21. Przecięcie zbiorów rozmytych A i B , gdy $\mu_A(x) < \mu_B(x)$



Rys. 4.22. Działanie operacji dopełnienia zbioru rozmytego

DEFINICJA 4.13

Dopełnieniem zbioru rozmytego $A \subseteq \mathbf{X}$ jest zbiór rozmyty \widehat{A} o funkcji przynależności

$$\mu_{\widehat{A}}(x) = 1 - \mu_A(x) \quad (4.59)$$

dla każdego $x \in \mathbf{X}$. Działanie operacji dopełnienia przedstawia rysunek 4.22.

Przykład 4.11

Założymy, że $\mathbf{X} = \{1, 2, 3, 4, 5, 6\}$ oraz

$$A = \frac{0,3}{2} + \frac{1}{3} + \frac{0,7}{5} + \frac{0,9}{6}. \quad (4.60)$$

Zgodnie z definicją 4.13, dopełnieniem zbioru A jest zbiór

$$\widehat{A} = \frac{1}{1} + \frac{0,7}{2} + \frac{1}{4} + \frac{0,3}{5} + \frac{0,1}{6}. \quad (4.61)$$

Zauważmy, że

$$A \cap \widehat{A} = \frac{0,3}{2} + \frac{0,3}{5} + \frac{0,1}{6} \neq \emptyset \quad (4.62)$$

oraz

$$A \cup \widehat{A} = \frac{1}{1} + \frac{0,7}{2} + \frac{1}{3} + \frac{1}{4} + \frac{0,7}{5} + \frac{0,9}{6} \neq \mathbf{X}. \quad (4.63)$$

Można wykazać, że przedstawione powyżej operacje na zbiorach rozmytych (definicje 4.11–4.13) mają cechy przemienności, łączności i rozdzielności, a ponadto zachodzą również prawa de Morgana oraz absorpcji. Jednakże w przypadku zbiorów rozmytych nie są spełnione prawa dopełnienia, tzn.

$$A \cap \widehat{A} \neq \emptyset, \quad (4.64)$$

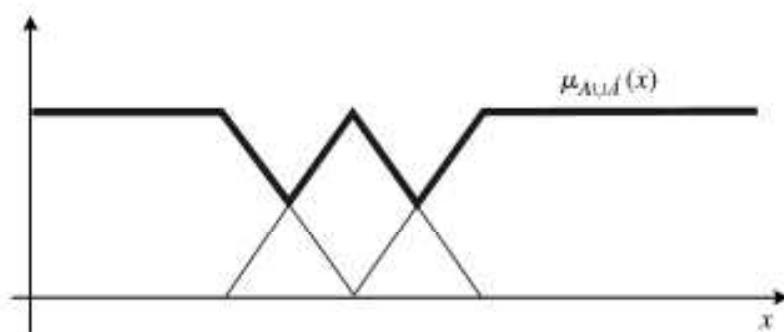
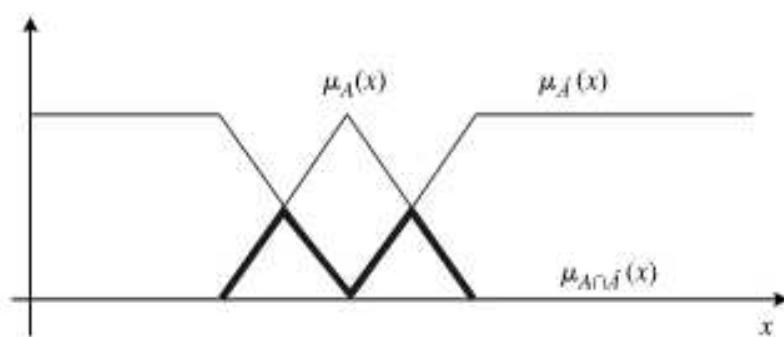
$$A \cup \widehat{A} \neq \mathbf{X}. \quad (4.65)$$

Fakt ten jest zilustrowany na rysunku 4.23 oraz w przykładzie 4.11. Warto wspomnieć, że funkcja przynależności przecięcia zbiorów rozmytych A i \widehat{A} spełnia nierówność (patrz [42]):

$$\mu_{A \cap \widehat{A}}(x) = \min(\mu_A(x), \mu_{\widehat{A}}(x)) \leq \frac{1}{2}. \quad (4.66)$$

Podobnie w przypadku sumy mamy

$$\mu_{A \cup \widehat{A}}(x) = \max(\mu_A(x), \mu_{\widehat{A}}(x)) \geq \frac{1}{2}. \quad (4.67)$$



Rys. 4.23. Zbiory rozmyte $A \cap \widehat{A}$ oraz $A \cup \widehat{A}$

DEFINICJA 4.14

Iloczyn kartezjański zbiorów rozmytych $A \subseteq \mathbf{X}$ i $B \subseteq \mathbf{Y}$ oznaczamy $A \times B$ i definiujemy jako

$$\mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y)) \quad (4.68)$$

lub

$$\mu_{A \times B}(x, y) = \mu_A(x)\mu_B(y) \quad (4.69)$$

dla każdego $x \in \mathbf{X}$ i $y \in \mathbf{Y}$. Iloczyn kartezjański zbiorów rozmytych $A_1 \subseteq \mathbf{X}_1$, $A_2 \subseteq \mathbf{X}_2, \dots, A_n \subseteq \mathbf{X}_n$ oznaczamy $A_1 \times A_2 \times \dots \times A_n$ i definiujemy jako

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n)) \quad (4.70)$$

lub

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \mu_{A_1}(x_1) \mu_{A_2}(x_2) \dots \mu_{A_n}(x_n) \quad (4.71)$$

dla każdego $x_1 \in \mathbf{X}_1$, $x_2 \in \mathbf{X}_2, \dots, x_n \in \mathbf{X}_n$.

Przykład 4.12

Założmy, że $\mathbf{X} = \{2, 4\}$, $\mathbf{Y} = \{2, 4, 6\}$ oraz

$$A = \frac{0,5}{2} + \frac{0,9}{4}, \quad (4.72)$$

$$B = \frac{0,3}{2} + \frac{0,7}{4} + \frac{0,1}{6}. \quad (4.73)$$

Stosując definicję 4.14 iloczynu kartezjańskiego zbiorów A i B , otrzymujemy

$$A \times B = \frac{0,3}{(2,2)} + \frac{0,5}{(2,4)} + \frac{0,1}{(2,6)} + \frac{0,3}{(4,2)} + \frac{0,7}{(4,4)} + \frac{0,1}{(4,6)}. \quad (4.74)$$

Następujące operacje algebraiczne na zbiorach rozmytych odgrywają znaczną rolę w semantyce zmiennych lingwistycznych (podrozdz. 4.8).

DEFINICJA 4.15

Koncentrację zbioru rozmytego $A \subseteq \mathbf{X}$ oznaczamy $CON(A)$ i definiujemy jako

$$\mu_{CON(A)}(x) = (\mu_A(x))^2 \quad (4.75)$$

dla każdego $x \in \mathbf{X}$.

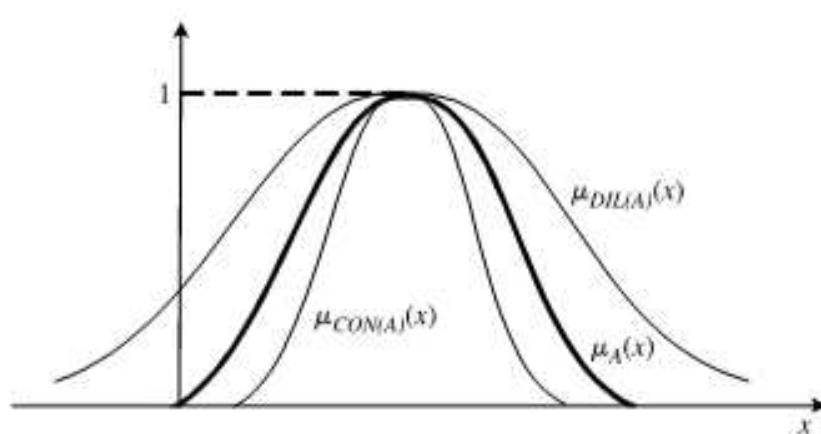
DEFINICJA 4.16

Rozcieńczenie zbioru rozmytego $A \subseteq \mathbf{X}$ oznaczamy $DIL(A)$ i definiujemy jako

$$\mu_{DIL(A)}(x) = (\mu_A(x))^{0,5} \quad (4.76)$$

dla każdego $x \in \mathbf{X}$.

Działanie operacji koncentracji i rozcieńczenia zbiorów rozmytych ilustruje rysunek 4.24.



Rys. 4.24. Działanie operacji koncentracji i rozcieńczenia zbioru rozmytego

Przykład 4.13

Jeżeli $\mathbf{X} = \{1, 2, 3, 4\}$ oraz

$$A = \frac{0,4}{2} + \frac{0,7}{3} + \frac{1}{4}, \quad (4.77)$$

to zgodnie z definicjami 4.15 oraz 4.16 mamy

$$CON(A) = \frac{0,16}{2} + \frac{0,49}{3} + \frac{1}{4}, \quad (4.78)$$

$$DIL(A) = \frac{0,63}{2} + \frac{0,84}{3} + \frac{1}{4}. \quad (4.79)$$

4.4. Zasada rozszerzania

Zasada rozszerzania pozwala przenieść (rozszerzyć) różne operacje matematyczne ze zbiorów nieroźmytych na zbiory rozmyte. Rozważmy pewne nieroźmyte odwzorowanie f przestrzeni \mathbf{X} w przestrzeń \mathbf{Y}

$$f : \mathbf{X} \rightarrow \mathbf{Y}. \quad (4.80)$$

Niech A będzie danym zbiorem rozmytym określonym w przestrzeni \mathbf{X} , tzn. $A \subseteq \mathbf{X}$. Jeżeli zbiór rozmyty A jest postaci (4.3), tzn.

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots + \frac{\mu_A(x_n)}{x_n}$$

oraz odwzorowanie f jest wzajemne jednoznaczne, to zasada rozszerzania mówi, że zbiór rozmyty B indukowany przez to odwzorowanie i określony w przestrzeni \mathbf{Y} jest postaci

$$B = f(A) = \frac{\mu_A(x_1)}{f(x_1)} + \frac{\mu_A(x_2)}{f(x_2)} + \dots + \frac{\mu_A(x_n)}{f(x_n)}. \quad (4.81)$$

Przykład 4.14

Założmy, że

$$A = \frac{0,1}{3} + \frac{0,4}{2} + \frac{0,7}{5} \quad (4.82)$$

oraz $f(x) = 2x + 1$. Zgodnie z zasadą rozszerzania mamy

$$B = f(A) = \frac{0,1}{7} + \frac{0,4}{5} + \frac{0,7}{11}. \quad (4.83)$$

Rozważmy teraz sytuację, w której więcej niż jeden element zbioru \mathbf{X} jest odwzorowany w ten sam element $y \in \mathbf{Y}$ (odwzorowanie f nie jest wzajemnie jednoznaczne). Wtedy stopień przynależności elementu y do zbioru rozmytego $B = f(A)$ jest równy maksymalnemu ze stopni przynależności elementów zbioru \mathbf{X} , które są odwzorowane w ten sam element y . Dla ilustracji tego przypadku zasady rozszerzania podamy następujący przykład.

Przykład 4.15

Jeżeli

$$A = \frac{0,3}{-2} + \frac{0,5}{3} + \frac{0,7}{2} \quad (4.84)$$

oraz $f(x) = x^2$, to zbiór rozmyty B indukowany przez odwzorowanie f jest równy

$$B = f(A) = \frac{0,5}{9} + \frac{0,7}{4}, \quad (4.85)$$

gdyż $\max\{0,3; 0,7\} = 0,7$. Oznaczmy przez $f^{-1}(y)$ zbiór tych elementów $x \in \mathbf{X}$, które są odwzorowane w element $y \in \mathbf{Y}$ poprzez przekształcenie f . Jeżeli $f^{-1}(y)$ jest zbiorem pustym, tzn. $f^{-1}(y) = \emptyset$, to stopień przynależności elementu y do zbioru rozmytego B jest równy zero. Powyższe rozważania oraz ilustrujące je przykłady pozwalają sformułować następującą zasadę rozszerzania.

Zasada rozszerzania I

Jeżeli mamy pewne nierożmyte odwzorowanie (4.80) i pewien zbiór rozmyty $A \subseteq \mathbf{X}$, to zasada rozszerzania mówi, że zbiór rozmyty B indukowany przez to odwzorowanie jest postaci

$$B = f(A) = \{(y, \mu_B(y)) \mid y = f(x), x \in \mathbf{X}\}, \quad (4.86)$$

gdzie

$$\mu_B(y) = \begin{cases} \sup_{x \in f^{-1}(y)} \mu_A(x), & \text{jeżeli } f^{-1}(y) \neq \emptyset, \\ 0, & \text{jeżeli } f^{-1}(y) = \emptyset. \end{cases} \quad (4.87)$$

Zasada rozszerzania I obejmuje przypadek zarówno przestrzeni \mathbf{X} o skończonej liczbie elementów (zbiór B jest określony wzorem (4.81)), jak i o nieskończonej liczbie elementów. W tym drugim przypadku zbiór rozmyty B indukowany przez odwzorowanie f można zapisać jako

$$B = f(A) = \int_y \frac{\mu_A(x)}{f(x)}. \quad (4.88)$$

W niektórych zastosowaniach (np. liczby rozmyte, podrozdział 4.5) przydatna jest inna postać zasady rozszerzania.

Zasada rozszerzania II

Niech \mathbf{X} będzie iloczynem kartezjańskim zbiorów nierożmytych $\mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n$. Jeżeli mamy pewne nierożmyte odwzorowanie

$$f : \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n \rightarrow \mathbf{Y} \quad (4.89)$$

oraz pewne zbiory rozmyte $A_1 \subseteq \mathbf{X}_1, A_2 \subseteq \mathbf{X}_2, \dots, A_n \subseteq \mathbf{X}_n$, to zasada rozszerzania mówi, że zbiór rozmyty B indukowany przez odwzorowanie f jest postaci

$$B = f(A_1, \dots, A_n) = \{(y, \mu_B(y)) \mid y = f(x_1, \dots, x_n), (x_1, \dots, x_n) \in \mathbf{X}\}, \quad (4.90)$$

przy czym

$$\mu_B(y) = \begin{cases} \sup_{(x_1, \dots, x_n) \in f^{-1}(y)} \min\{\mu_{A_1}(x_1), \dots, \mu_{A_n}(x_n)\}, & \text{jeżeli } f^{-1}(y) \neq \emptyset, \\ 0, & \text{jeżeli } f^{-1}(y) = \emptyset. \end{cases} \quad (4.91)$$

We wzorze (4.91) operacja minimum może być zastąpiona iloczynem algebraicznym lub ogólniej tzw. t -normą (podrozdz. 4.6). Kolejne trzy przykłady ilustrują fakt, że zasada rozszerzania pozwala przenosić operacje arytmetyczne na zbiory rozmyte.

Przykład 4.16

Rozważmy funkcję

$$f(x_1, x_2) = \frac{x_1 x_2}{(x_1 + x_2)}. \quad (4.92)$$

Wyznaczamy zbiór rozmyty $B = f(A_1, A_2)$ indukowany przez odwzorowanie (4.92). Zgodnie ze wzorem (4.90) mamy

$$B = f(A_1, A_2) = \int_{x_1 \in \mathbf{X}_1} \int_{x_2 \in \mathbf{X}_2} \sup_{(x_1, \dots, x_n \in f^{-1}(y))} \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2)) \left| \frac{x_1 x_2}{x_1 + x_2} \right|$$

Przykład 4.17

Założymy, że \mathbf{X} jest iloczynem kartezjańskim zbiorów $\mathbf{X}_1 = \mathbf{X}_2 = \{1, 2, 3, 4, 5, 6\}$. Niech A_1 będzie zbiorem rozmytym liczb „bliskich liczby 2”

$$A_1 = \frac{0,7}{1} + \frac{1}{2} + \frac{0,8}{3} \quad (4.93)$$

oraz niech A_2 będzie zbiorem rozmytym liczb „bliskich liczby 4”

$$A_2 = \frac{0,8}{3} + \frac{1}{4} + \frac{0,9}{5}. \quad (4.94)$$

Jeżeli

$$y = f(x_1, x_2) = x_1 x_2, \quad (4.95)$$

to zbiór $B = f(A_1, A_2)$ indukowany przez odwzorowanie (4.95) będzie zbiorem rozmytym liczb „bliskich liczby 8”, przy czym $B \subseteq \mathbf{Y} = \{1, 2, \dots, 36\}$. Na mocy zasady rozszerzania II mamy

$$\begin{aligned} B = f(A_1, A_2) &= \sum_{i,j=1}^3 [\min(\mu_{A_1}(x_1^{(i)}), \mu_{A_2}(x_2^{(j)}))] / x_1^{(i)} x_2^{(j)} \\ &= \frac{\min(0,7; 0,8)}{3} + \frac{\min(0,7; 1)}{4} + \frac{\min(0,7; 0,9)}{5} + \frac{\min(1; 0,8)}{6} + \frac{\min(1; 1)}{8} \\ &\quad + \frac{\min(1; 0,9)}{10} + \frac{\min(0,8; 0,8)}{9} + \frac{\min(0,8; 1)}{12} + \frac{\min(0,8; 0,9)}{15} \\ &= \frac{0,7}{3} + \frac{0,7}{4} + \frac{0,7}{5} + \frac{0,8}{6} + \frac{1}{8} + \frac{0,8}{9} + \frac{0,9}{10} + \frac{0,8}{12} + \frac{0,8}{15}. \end{aligned} \quad (4.96)$$

Następny przykład ilustruje przypadek, gdy element $y = f(x_1^{(i)}, x_2^{(j)})$ przyjmuje tę samą wartość dla różnych wartości elementów $x_1^{(i)}$ i $x_2^{(j)}$.

Przykład 4.18

Założymy, że \mathbf{X} jest iloczynem kartezjańskim zbiorów $\mathbf{X}_1 = \mathbf{X}_2 = \{1, 2, 3, 4\}$. Zdefiniujmy następujący zbiór rozmyty A_1 liczb „bliskich liczby 2”

$$A_1 = \frac{0,7}{1} + \frac{1}{2} + \frac{0,8}{3} \quad (4.97)$$

oraz zbiór rozmyty A_2 liczb „bliskich liczby 3”

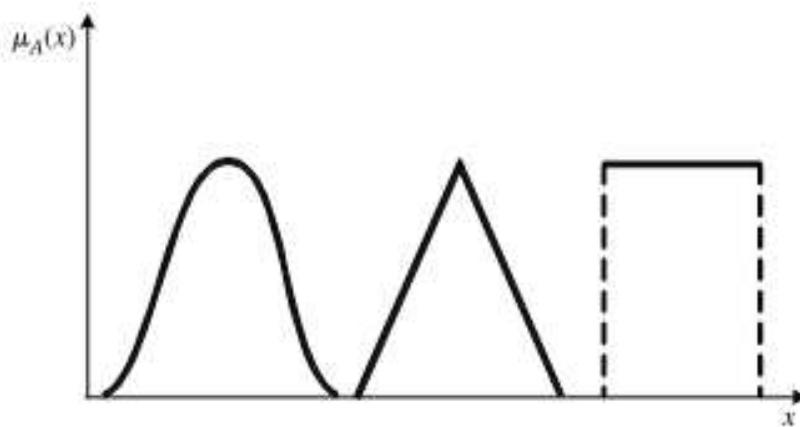
$$A_2 = \frac{0,8}{2} + \frac{1}{3} + \frac{0,6}{4}. \quad (4.98)$$

Obecnie zbiór $B = f(A_1, A_2)$ indukowany przez odwzorowanie (4.95) będzie zbiorem rozmytym liczb „bliskich liczby 6”, przy czym $B \subseteq \mathbf{Y} = \{1, 2, \dots, 16\}$. Zgodnie z zasadą rozszerzania II mamy

$$\begin{aligned} B = f(A_1, A_2) &= \frac{\min(0,7; 0,8)}{2} + \frac{\min(0,7; 1)}{3} \\ &+ \frac{\max[\min(0,7; 0,6); \min(1; 0,8)]}{4} + \frac{\max[\min(1; 1); \min(0,8; 0,8)]}{6} \\ &+ \frac{\min(1; 0,6)}{8} + \frac{\min(0,8; 1)}{9} + \frac{\min(0,8; 0,6)}{12} \\ &= \frac{0,7}{2} + \frac{0,7}{3} + \frac{0,8}{4} + \frac{1}{6} + \frac{0,6}{8} + \frac{0,8}{9} + \frac{0,6}{12}. \end{aligned} \quad (4.99)$$

4.5. Liczby rozmyte

W teorii zbiorów rozmytych wyróżnia się zbiory rozmyte, które są zdefiniowane na osi liczb rzeczywistych. Na przykład zbiory rozmyte liczb „bliskich liczby 7” (rys. 4.25) są zdefiniowane w zbiorze \mathbf{R} , a ponadto są normalne i wypukłe oraz mają ciągłą funkcję przynależności. Tego typu zbiory rozmyte nazywamy liczbami rozmytymi. Podamy teraz definicję liczby rozmytej.



Rys. 4.25. Przykłady liczb rozmytych

DEFINICJA 4.17

Zbiór rozmyty A określony w zbiorze liczb rzeczywistych, $A \subseteq \mathbf{R}$, którego funkcja przynależności

$$\mu_A : \mathbf{R} \rightarrow [0, 1]$$

spełnia warunki:

- 1) $\sup_{x \in \mathbf{R}} \mu_A(x) = 1$, tzn. zbiór rozmyty A jest normalny,

- 2) $\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min\{\mu_A(x_1), \mu_A(x_2)\}$, tzn. zbiór A jest wypukły,
 3) $\mu_A(x)$ jest funkcją przedziałami ciągłą.

nazywamy *liczbą rozmytą*.

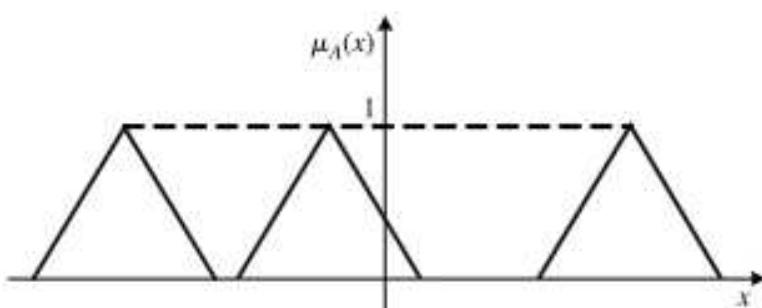
Na rysunku 4.25 przedstawiono przykłady liczb rozmytych. W teorii liczb rozmytych rozróżnia się liczby rozmyte dodatnie i ujemne.

DEFINICJA 4.18

Liczba rozmyta $A \subseteq \mathbf{R}$ jest *dodatnia*, jeżeli $\mu_A(x) = 0$ dla wszystkich $x < 0$.

Liczba rozmyta $A \subseteq \mathbf{R}$ jest ujemna, jeżeli $\mu_A(x) = 0$ dla wszystkich $x > 0$.

Na rysunku 4.26 przedstawiono przykład liczby rozmytej dodatniej, ujemnej oraz liczby, która nie jest ani dodatnia, ani ujemna.



Rys. 4.26. Przykład liczby rozmytej dodatniej, ujemnej oraz liczby, która nie jest ani dodatnia, ani ujemna

Czytelnik, który zapoznał się z treścią podrozdziału 4.4, nie będzie miał trudności ze zdefiniowaniem podstawowych operacji arytmetycznych na liczbach rozmytych. Operacje te określmy za pomocą zasady rozszerzania, która pozwala sformułować definicję dodawania, odejmowania, mnożenia i dzielenia dwóch liczb rozmytych $A_1, A_2 \subseteq \mathbf{R}$.

Definicja 4.19 jest konsekwencją zasady rozszerzania II, w której odwzorowanie (4.89) przybiera postać

$$y = f(x_1, x_2) = \begin{cases} x_1 + x_2 & \text{w przypadku dodawania} \\ x_1 - x_2 & \text{w przypadku odejmowania} \\ x_1 \cdot x_2 & \text{w przypadku mnożenia} \\ x_1 : x_2 & \text{w przypadku dzielenia} \end{cases} \quad \begin{array}{l} \text{liczb rozmytych } A_1 \text{ i } A_2, \\ \text{liczb rozmytych } A_1 \text{ i } A_2, \\ \text{liczb rozmytych } A_1 \text{ i } A_2, \\ \text{liczb rozmytych } A_1 \text{ i } A_2. \end{array}$$

DEFINICJA 4.19

Podstawowe operacje arytmetyczne na liczbach rozmytych $A_1, A_2 \subseteq \mathbf{R}$ określamy następująco:

a) Dodawanie dwóch liczb rozmytych A_1 i A_2 oznaczamy

$$A_1 \oplus A_2 \stackrel{\text{def}}{=} B, \quad (4.100)$$

przy czym funkcja przynależności sumy (4.100) jest określona wzorem (4.91) przybierającym postać

$$\mu_B(y) = \sup_{\substack{x_1, x_2 \\ y = x_1 + x_2}} \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2)\}. \quad (4.101)$$

b) *Odejmowanie* dwu liczb rozmytych A_1 i A_2 oznaczamy

$$A_1 \ominus A_2 \stackrel{\text{def}}{=} B, \quad (4.102)$$

przy czym funkcja przynależności różnicy (4.102) jest określona wzorem (4.91) przybierającym postać

$$\mu_B(y) = \sup_{\substack{x_1, x_2 \\ y=x_1-x_2}} \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2)\}. \quad (4.103)$$

c) *Mnożenie* dwu liczb rozmytych A_1 i A_2 oznaczamy

$$A_1 \odot A_2 \stackrel{\text{def}}{=} B, \quad (4.104)$$

przy czym funkcja przynależności iloczynu (4.104) jest określona wzorem (4.91) przybierającym postać

$$\mu_B(y) = \sup_{\substack{x_1, x_2 \\ y=x_1 \cdot x_2}} \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2)\}. \quad (4.105)$$

d) *Dzielenie* dwu liczb rozmytych A_1 i A_2 oznaczamy

$$A_1 \odot A_2 \stackrel{\text{def}}{=} B, \quad (4.106)$$

przy czym funkcja przynależności ilorazu (4.106) jest określona wzorem (4.91) przybierającym postać

$$\mu_B(y) = \sup_{\substack{x_1, x_2 \\ y=x_1 : x_2}} \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2)\}. \quad (4.107)$$

Jakkolwiek z punktu widzenia zastosowań interesują nas liczby rozmyte mające ciągłe funkcje przynależności, to dla ilustracji powyższej definicji rozważymy przypadek dyskretny.

Przykład 4.19

Dodamy oraz pomnożymy dwie liczby rozmyte postaci

$$A_1 = \frac{0.7}{2} + \frac{1}{3} + \frac{0.6}{4}, \quad (4.108)$$

$$A_2 = \frac{0.8}{3} + \frac{1}{4} + \frac{0.5}{6}. \quad (4.109)$$

Zgodnie ze wzorem (4.101) mamy

$$\begin{aligned} A_1 \oplus A_2 &= \frac{\min(0.7; 0.8)}{5} + \frac{\max\{\min(0.7; 1), \min(1; 0.8)\}}{6} \\ &+ \frac{\max\{\min(1; 1), \min(0.6; 0.8)\}}{7} + \frac{\max\{\min(0.7; 0.5), \min(0.6; 1)\}}{8} \\ &+ \frac{\min(1; 0.5)}{9} + \frac{\min(0.6; 0.5)}{10} \\ &= \frac{0.7}{5} + \frac{0.8}{6} + \frac{1}{7} + \frac{0.6}{8} + \frac{0.5}{9} + \frac{0.5}{10}. \end{aligned} \quad (4.110)$$

Na podstawie wzoru (4.105) otrzymujemy

$$\begin{aligned}
 A_1 \odot A_2 &= \frac{\min(0,7; 0,8)}{6} + \frac{\min(0,7; 1)}{8} + \frac{\min(1; 0,8)}{9} \\
 &\quad + \frac{\max\{\min(0,7; 0,5), \min(1; 1), \min(0,6; 0,8)\}}{12} \\
 &\quad + \frac{\min(0,6; 1)}{16} + \frac{\min(1; 0,5)}{18} + \frac{\min(0,6; 0,5)}{24} \\
 &= \frac{0,7}{6} + \frac{0,7}{8} + \frac{0,8}{9} + \frac{1}{12} + \frac{0,6}{16} + \frac{0,5}{18} + \frac{0,5}{24}.
 \end{aligned} \tag{4.111}$$

Nie zawsze wynikiem operacji arytmetycznych na liczbach rozmytych jest liczba rozmyta. Problem ten zostaje wyeliminowany, gdy przeprowadzamy operacje na liczbach rozmytych mających ciągłe funkcje przynależności, o czym mówi następujące twierdzenie.

TWIERDZENIE 4.2 (Dubois i Prade [42])

Jeżeli liczby rozmyte A_1 i A_2 mają ciągłe funkcje przynależności, to wynikiem operacji arytmetycznych dodawania, odejmowania, mnożenia i dzielenia są liczby rozmyte.

Omówiliśmy podstawowe operacje dwuargumentowe na liczbach rozmytych. Operacje jednoargumentowe przeprowadza się również za pomocą zasady rozszerzania. Jeżeli f jest odwzorowaniem

$$f : \mathbf{R} \rightarrow \mathbf{R} \tag{4.112}$$

oraz $A \subseteq \mathbf{R}$, $y = f(x)$, to zgodnie ze wzorem (4.87) mamy

$$\mu_B(y) = \sup_{\substack{x \\ y=f(x)}} \mu_A(x), \tag{4.113}$$

gdzie $B = f(A)$.

Podamy teraz kilka przykładów operacji jednoargumentowych na liczbach rozmytych.

1) *Operacja zmiany znaku.* W wyniku operacji $f(x) = -x$ otrzymujemy liczbę rozmytą przeciwną do liczby rozmytej $A \subseteq \mathbf{R}$. Liczbę tę oznaczamy $-A \subseteq \mathbf{R}$, a jej funkcja przynależności jest równa

$$\mu_{-A}(x) = \mu_A(-x). \tag{4.114}$$

Liczby rozmyte A i $-A$ są symetryczne względem osi x .

2) *Operacja odwrotności.* W wyniku operacji $f(x) = x^{-1}$, $x \neq 0$, otrzymujemy liczbę rozmytą odwrotną do liczby rozmytej $A \subseteq \mathbf{R}$. Liczbę tę oznaczamy $A^{-1} \subseteq \mathbf{R}$, a jej funkcja przynależności jest równa

$$\mu_{A^{-1}}(x) = \mu_A(x^{-1}). \tag{4.115}$$

Zakładamy, że A jest liczbą rozmytą dodatnią lub ujemną. Jeżeli liczba rozmyta A nie jest ani dodatnia, ani ujemna, to zbiór rozmyty $B = f(A) = A^{-1}$ nie jest wypukły, a więc B nie jest liczbą rozmytą.

3) *Operacja skalowania.* W wyniku operacji $f(x) = \lambda x$, $\lambda \neq 0$, otrzymujemy liczbę rozmytą przeskalowaną w stosunku do liczby rozmytej $A \subseteq \mathbf{R}$. Liczbę tę oznaczamy $\lambda A \subseteq \mathbf{R}$, a jej funkcja przynależności jest równa

$$\mu_{\lambda A}(x) = \mu_A(x\lambda^{-1}). \tag{4.116}$$

4) *Operacja eksponent.* W wyniku operacji $f(x) = e^x$, $x > 0$, otrzymujemy potęgę liczby rozmytej $A \subseteq \mathbf{R}$. Liczbę tę oznaczamy $e^A \subseteq \mathbf{R}$, a jej funkcja przynależności jest równa

$$\mu_{e^A}(x) = \begin{cases} \mu_A(\log x) & \text{dla } x > 0, \\ 0 & \text{dla } x \leq 0, \end{cases} \quad (4.117)$$

a zatem e^A jest liczbą rozmytą dodatnią.

5) *Operacja wartości bezwzględnej.* Wartość bezwzględną liczby rozmytej $A \subseteq \mathbf{R}$ oznaczamy $|A| \subseteq \mathbf{R}$ i określamy jako

$$\mu_{|A|}(x) = \begin{cases} \max(\mu_A(x), \mu_A(-x)) & \text{dla } x \geq 0, \\ 0 & \text{dla } x < 0. \end{cases} \quad (4.118)$$

Oczywiście $|A|$ jest liczbą rozmytą dodatnią.

Przykład 4.20

Jeżeli

$$A = \frac{0,7}{1} + \frac{1}{2} + \frac{0,6}{5}, \quad (4.119)$$

to liczba rozmyta $-A$ ma postać

$$-A = \frac{0,6}{-5} + \frac{1}{-2} + \frac{0,7}{-1}, \quad (4.120)$$

natomiast liczbę rozmytą A^{-1} zapisujemy jako

$$A^{-1} = \frac{0,6}{0,2} + \frac{1}{0,5} + \frac{0,7}{1}. \quad (4.121)$$

Korzystając z definicji 4.19, łatwo sprawdzić, że w powyższym przykładzie

$$A + (-A) \neq \frac{1}{0} \quad (4.122)$$

oraz

$$A \cdot A^{-1} \neq \frac{1}{1}. \quad (4.123)$$

Zatem liczby rozmyte charakteryzują się brakiem liczby rozmytej przeciwej i odwrotnej względem dodawania i mnożenia. Fakt ten uniemożliwia np. zastosowanie metody eliminacji do rozwiązywania równań, w których występują liczby rozmyte.

Operacje arytmetyczne na liczbach rozmytych wymagają dość skomplikowanych obliczeń. Dlatego Dubois i Prade [41] zaproponowali pewną szczególną reprezentację liczb rozmytych. Reprezentacja ta przedstawia liczby rozmyte za pomocą 3 parametrów, co znacznie upraszcza wykonywanie operacji arytmetycznych. Niech L oraz P będą funkcjami odwzorowującymi

$$(-\infty, \infty) \rightarrow [0, 1] \quad (4.124)$$

oraz spełniającymi warunki

- 1) $L(-x) = L(x)$ oraz $P(-x) = P(x)$,
- 2) $L(0) = 1$ oraz $P(0) = 1$,
- 3) L i P są funkcjami nierośnającymi w przedziale $[0, +\infty)$.

Jako przykłady funkcji L można podać:

$$L(x) = P(x) = e^{-|x|^p} \quad p > 0, \quad (4.125)$$

$$L(x) = P(x) = \frac{1}{1 + |x|^p} \quad p > 0, \quad (4.126)$$

$$L(x) = P(x) = \max(0, 1 - |x|^p) \quad p > 0, \quad (4.127)$$

$$L(x) = P(x) = \begin{cases} 1 & \text{dla } x \in [-1, 1], \\ 0 & \text{dla } x \notin [-1, 1]. \end{cases} \quad (4.128)$$

Podamy teraz definicję liczby rozmytej typu $L-P$.

DEFINICJA 4.20

Liczba rozmyta $A \subseteq \mathbf{R}$ jest *liczbą rozmytą typu L-P* wtedy i tylko wtedy, gdy jej funkcja przynależności ma postać

$$\mu_A(x) = \begin{cases} L\left(\frac{m-x}{\alpha}\right), & \text{jeżeli } x \leq m, \\ P\left(\frac{x-m}{\beta}\right), & \text{jeżeli } x \geq m, \end{cases} \quad (4.129)$$

gdzie m to liczba rzeczywista, zwana wartością średnią liczby rozmytej A ($\mu_A(m) = 1$), α — liczba rzeczywista dodatnia, zwana rozrzutem lewostronnym, β — liczba rzeczywista dodatnia, zwana rozrzutem prawostronnym. Zauważmy, że jeżeli rozrzuty α i β zwiększają się, to liczba A staje się „bardziej” rozmyta. Liczbę rozmytą typu $L-P$ można krótko zapisać jako

$$A = (m_A, \alpha_A, \beta_A)_{LP}. \quad (4.130)$$

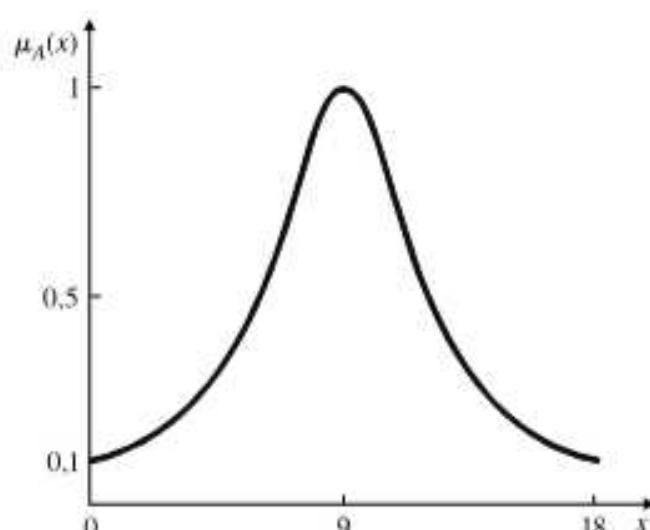
Przykład 4.21

Liczبę rozmytą „mniej więcej 9” można zapisać w postaci

$$A = (9, 3, 3)_{LP}. \quad (4.131)$$

Funkcję przynależności tej liczby przedstawia rysunek 4.27, przy czym

$$L(x) = P(x) = \frac{1}{1 + x^2}. \quad (4.132)$$



Rys. 4.27. Ilustracja do przykładu 4.21

Operacje arytmetyczne na liczbach rozmytych typu $L-P$ sprowadzają się do operacji na trzech parametrach. Liczba rozmyta przeciwna do liczby rozmytej (4.130) jest równa

$$-A = (-m_A, \alpha, \beta)_{LP}. \quad (4.133)$$

Suma liczb rozmytych $A = (m_A, \alpha_A, \beta_A)_{LP}$ i $B = (m_B, \alpha_B, \beta_B)_{LP}$ ma postać

$$A \oplus B = (m_A + m_B, \alpha_A + \alpha_B, \beta_A + \beta_B)_{LP}. \quad (4.134)$$

Inne operacje arytmetyczne (np. mnożenie i dzielenie) na liczbach rozmytych typu $L-P$ są bardziej skomplikowane, a ich wynik ma charakter przybliżony.

Funkcja przynależności $\mu_A(x)$ liczby rozmytej typu $L-P$ przyjmuje wartość 1 tylko w punkcie $x = m$. Zmodyfikujmy teraz definicję 4.20 tak, aby $\mu_A(x) = 1$ nie tylko w jednym punkcie $x = m$, ale we wszystkich punktach przedziału $[m_1, m_2]$, przy czym $m_1 < m_2$ oraz $m_1, m_2 \in \mathbf{R}$. Otrzymamy wówczas definicję tzw. płaskiej liczby rozmytej. Definicję tę można zastosować do modelowania przedziałów rozmytych.

DEFINICJA 4.21

Płaską liczbą rozmytą typu $L-P$ nazywamy liczbę rozmytą o funkcji przynależności

$$\mu_A(x) = \begin{cases} L\left(\frac{m_1 - x}{\alpha}\right), & \text{jeżeli } x \leq m_1, \\ 1, & \text{jeżeli } m_1 \leq x \leq m_2, \\ P\left(\frac{x - m_2}{\beta}\right), & \text{jeżeli } x \geq m_2. \end{cases} \quad (4.135)$$

Płaską liczbę rozmytą A możemy utożsamiać z przedziałem rozmytym A postaci

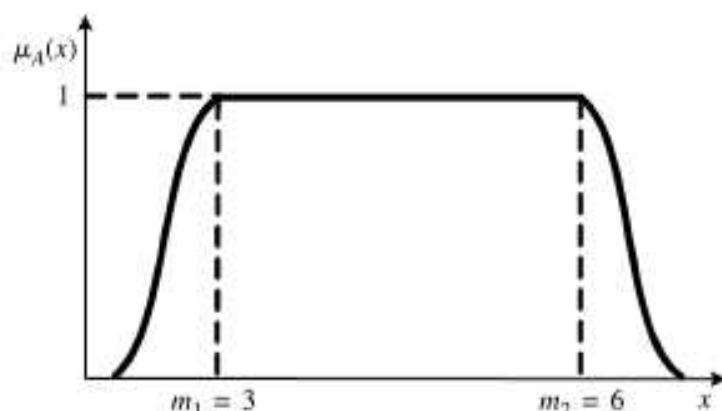
$$A = (m_1, m_2, \alpha, \beta)_{LP}. \quad (4.136)$$

Przykład 4.22

Rozważmy nieprecyzyjne stwierdzenie „cena roweru w tym sklepie waha się od ok. 3 do ok. 6 tys. zł”. Adekwatną formalizacją tego stwierdzenia może być przedział rozmyty A postaci

$$A = (3, 6, \alpha, \beta)_{LP}. \quad (4.137)$$

Na rysunku 4.28 przedstawiono przykładowy wykres funkcji przynależności przedziału rozmytego (4.137).



Rys. 4.28. Ilustracja do przykładu 4.22: przedział rozmyty „od ok. 3 do ok. 6 tys. zł”

W teorii liczb rozmytych na szczególną uwagę zasługują trójkątne liczby rozmyte. Możemy je opisać funkcjami przynależności klasy t (4.21). Obecnie zaproponujemy inny opis tych liczb. *Trójkątna* liczba rozmyta A jest zdefiniowana w przedziale $[a_1, a_2]$, a jej funkcja przynależności przyjmuje wartość równą 1 w punkcie a_M . Zatem trójkątną liczbę rozmytą możemy zapisać w symbolicznej postaci

$$A = (a_1, a_M, a_2). \quad (4.138)$$

W wielu zastosowaniach zachodzi potrzeba tzw. *wyostrzenia* trójkątnej liczby rozmytej. W wyniku wyostrzenia wiedza opisana przez liczbę rozmytą zostaje przedstawiona w postaci liczby rzeczywistej. Poniżej przedstawiamy cztery sposoby wyostrzania trójkątnej liczby rozmytej [13]:

$$\begin{aligned} y^{(1)} &= a_M, \\ y^{(2)} &= \frac{a_1 + a_M + a_2}{3}, \\ y^{(3)} &= \frac{a_1 + 2a_M + a_2}{4}, \\ y^{(4)} &= \frac{a_1 + 4a_M + a_2}{6}. \end{aligned}$$

Zauważmy, że wartości a_1 i a_2 nie mają wpływu na wyznaczenie wartości wyostrzonej $y^{(1)}$. Warto wspomnieć, że suma dwóch trójkątnych liczb rozmytych $A_1 = (a_1^{(1)}, a_M^{(1)}, a_2^{(1)})$ i $A_2 = (a_1^{(2)}, a_M^{(2)}, a_2^{(2)})$ jest również liczbą trójkątną [171]

$$\begin{aligned} A_1 + A_2 &= (a_1^{(1)}, a_M^{(1)}, a_2^{(1)}) + (a_1^{(2)}, a_M^{(2)}, a_2^{(2)}) \\ &= (a_1^{(1)} + a_1^{(2)}, a_M^{(1)} + a_M^{(2)}, a_2^{(1)} + a_2^{(2)}). \end{aligned} \quad (4.139)$$

Właściwość ta również zachodzi dla $n > 2$ liczb trójkątnych.

4.6. Normy trójkątne i negacje

W podrozdziale 4.3 zdefiniowaliśmy operacje przecięcia i sumy zbiorów rozmytych jako

$$\begin{aligned} \mu_{A \cap B}(x) &= \min(\mu_A(x), \mu_B(x)), \\ \mu_{A \cup B}(x) &= \max(\mu_A(x), \mu_B(x)). \end{aligned}$$

Jednocześnie podkreśliliśmy, że nie są to jedyne definicje tych operacji. *Przecięcie* zbiorów rozmytych możemy zdefiniować ogólniej jako

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)), \quad (4.140)$$

gdzie funkcja T jest tzw. t -normą. Zatem $\min(\mu_A(x), \mu_B(x)) = T(\mu_A(x), \mu_B(x))$ jest przykładem działania t -normy. Podobnie sumę zbiorów rozmytych definiujemy następująco:

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)), \quad (4.141)$$

gdzie funkcja S jest tzw. t -konormą. W tym przypadku $\max(\mu_A(x), \mu_B(x)) = S(\mu_A(x), \mu_B(x))$ jest przykładem działania t -konormy. Warto wspomnieć, że t -normy oraz t -konormy należą do tzw. norm trójkątnych. Normy te będziemy stosować wielokrotnie w dalszej części tej książki, nie tylko do definiowania operacji przecięcia i sumy zbiorów rozmytych.

Poznaliśmy przykłady działania t -normy oraz t -konormy, a teraz przedstawimy ich formalne definicje.

DEFINICJA 4.22

Funkcję dwóch zmiennych T

$$T : [0, 1] \times [0, 1] \rightarrow [0, 1] \quad (4.142)$$

nazywamy *t-normą*, jeżeli

(i) funkcja T jest niemalejąca względem obu argumentów

$$T(a, c) \leq T(b, d) \quad \text{dla } a \leq b, c \leq d; \quad (4.143)$$

(ii) funkcja T spełnia warunek przemienności

$$T(a, b) = T(b, a); \quad (4.144)$$

(iii) funkcja T spełnia warunek łączności

$$T(T(a, b), c) = T(a, T(b, c)); \quad (4.145)$$

(iv) funkcja T spełnia warunek brzegowy

$$T(a, 1) = a, \quad (4.146)$$

gdzie $a, b, c, d \in [0, 1]$.

Z założeń mamy

$$T(a, 0) = T(0, a) \leq T(0, 1) = 0. \quad (4.147)$$

Zatem drugi warunek brzegowy przybiera postać

$$T(a, 0) = 0. \quad (4.148)$$

W dalszej części rozdziału działanie t -normy na argumentach a i b będziemy oznaczać

$$T(a, b) = a^T b. \quad (4.149)$$

Jeżeli np. a i b utożsamiamy z funkcjami przynależności zbiorów rozmytych A i B , to równość (4.140) zapisujemy jako

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x)^T \mu_B(x). \quad (4.150)$$

Korzystając z właściwości (4.145), definicję t -normy można w sposób następujący uogólnić na przypadek t -normy wielu zmiennych

$$\bigcap_{i=1}^n \{a_i\} = T \left\{ \bigcap_{i=1}^{n-1} \{a_i\}, a_n \right\} = T\{a_1, a_2, \dots, a_n\} = T\{\mathbf{a}\} = a_1^T a_2^T \dots a_n^T. \quad (4.151)$$

Poniżej przedstawiono definicję normy trójkątnej z wagami $0 \leq w_i \leq 1$, $i = 1, \dots, n$.

DEFINICJA 4.23

Ważoną t -normę oznaczamy jako T^* i definiujemy w sposób następujący:

$$T^*\{a_1, \dots, a_n; w_1, \dots, w_n\} = \bigwedge_{i=1}^n \{1 - w_i(1 - a_i)\}, \quad (4.152)$$

gdzie T jest dowolną t -normą, natomiast wagi spełniają warunek $0 \leq w_i \leq 1$, $i = 1, \dots, n$.

Założymy, że $w_i = 1$, $i = 1, \dots, n$. Wówczas ważona t -norma T^* redukuje się do t -normy T . W rozdziałach 9 i 10 wagi w_i interpretujemy jako stopnie prawdziwości poprzedników rozmytych reguł lub stopnie prawdziwości poszczególnych reguł w tzw. modelu logicznym. Łatwo sprawdzić, że ważona t -norma (podobnie jak ważona t -konorma — definicja 4.25) nie spełnia warunków brzegowych klasycznej t -normy. Jednakże, jak pokażemy w rozdziałach 9 i 10, zastosowanie tej koncepcji pozwala zaprojektować struktury neuronowo-rozmyte charakteryzujące się bardzo małym błędem działania.

DEFINICJA 4.24

Funkcję dwóch zmiennych S

$$S : [0, 1] \times [0, 1] \rightarrow [0, 1] \quad (4.153)$$

nazywamy t -konormą, jeżeli jest niemalejąca względem obu argumentów, spełnia warunek przemienności i łączności, a ponadto spełniony jest następujący warunek brzegowy:

$$S(a, 0) = a. \quad (4.154)$$

Z założeń oraz warunku (4.154) wynika, że

$$S(a, 1) = S(1, a) \geq S(1, 0) = 1. \quad (4.155)$$

Zatem drugi warunek brzegowy przybiera postać

$$S(a, 1) = 1. \quad (4.156)$$

Działanie t -konormy na argumentach a i b będziemy oznaczać

$$S(a, b) = a \overset{S}{*} b. \quad (4.157)$$

Korzystając z właściwości łączności, powyższą definicję można w sposób następujący uogólnić na przypadek t -konormy wielu zmiennych:

$$\bigwedge_{i=1}^n \{a_i\} = T \left\{ \bigwedge_{i=1}^{n-1} \{a_i\}, a_n \right\} = S\{a_1, a_2, \dots, a_n\} = S\{\mathbf{a}\} = a_1 \overset{S}{*} a_2 \overset{S}{*} \dots \overset{S}{*} a_n. \quad (4.158)$$

DEFINICJA 4.25

Ważoną t -konormę oznaczamy jako S^* i definiujemy następująco:

$$S^*\{a_1, \dots, a_n; w_1, \dots, w_n\} = \bigwedge_{i=1}^n \{w_i a_i\}. \quad (4.159)$$

W rozdziałach 9 i 10 wagi w_i interpretujemy jako stopnie prawdziwości poszczególnych reguł w tzw. modelu Mamdaniego.

DEFINICJA 4.26

Funkcje T i S , spełniające warunki

$$\sum_{i=1}^n S\{a_i\} = 1 - \sum_{i=1}^n T\{1 - a_i\}, \quad (4.160)$$

$$\sum_{i=1}^n T\{a_i\} = 1 - \sum_{i=1}^n S\{1 - a_i\}, \quad (4.161)$$

nazywamy normami trójkątnymi *dualnymi*.

DEFINICJA 4.27

Mówimy, że para dualnych operatorów trójkątnych T i S ma właściwość archimedesową, jeżeli

$$T\{a, a\} < a < S\{a, a\} \quad (4.162)$$

dla wszystkich $a \in (0, 1)$.

DEFINICJA 4.28

Mówimy, że para dualnych ciągłych operatorów trójkątnych T i S jest typu np (*nilpotent*), jeżeli dla dowolnego ciągu argumentów $a_i \in (0, 1)$, $i = 1, 2, \dots$, istnieje wskaźnik n taki, że

$$T\{a_1, a_2, \dots, a_n\} = 0, \quad (4.163)$$

$$S\{a_1, a_2, \dots, a_n\} = 1. \quad (4.164)$$

DEFINICJA 4.29

Mówimy, że para dualnych ciągłych operatorów trójkątnych T i S jest typu st (*strict*), jeżeli

$$T\{a_1, a_2, \dots, a_n\} > 0, \quad (4.165)$$

$$S\{a_1, a_2, \dots, a_n\} < 1 \quad (4.166)$$

dla $0 < a_i < 1$, $i = 1, \dots, n$, $n \geq 2$ oraz $a_1 = a_2 = \dots = a_n$.

Poniżej przedstawione zostały przykłady niektórych t -norm i odpowiadających im t -konorm.

Przykład 4.23 (Normy trójkątne typu min/max)

Normy trójkątne typu min/max, zwane normami trójkątnymi Zadeha, są opisane poniższymi zależnościami:

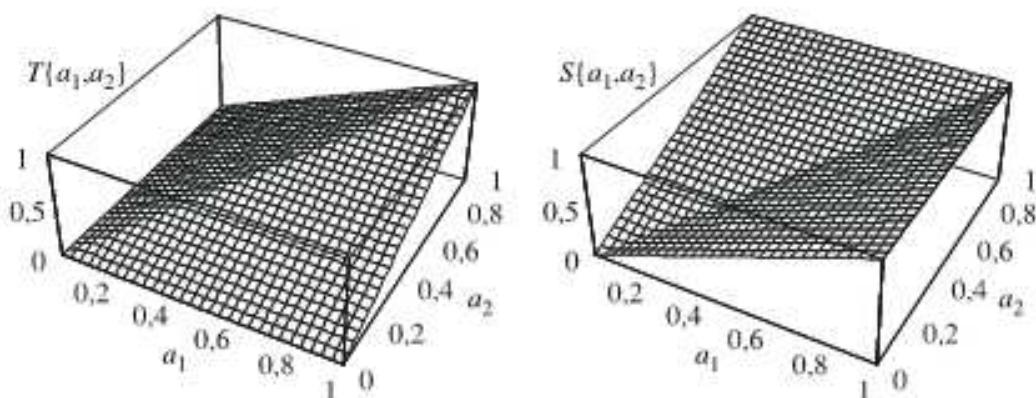
$$T_M\{a_1, a_2\} = \min\{a_1, a_2\}, \quad (4.167)$$

$$S_M\{a_1, a_2\} = \max\{a_1, a_2\}, \quad (4.168)$$

$$T_M\{a_1, a_2, \dots, a_n\} = \min_{i=1, \dots, n} \{a_i\}, \quad (4.169)$$

$$S_M\{a_1, a_2, \dots, a_n\} = \max_{i=1, \dots, n} \{a_i\}. \quad (4.170)$$

Normy trójkątne min/max są dualne, lecz nie są archimedesowe.



Rys. 4.29. Kształt hiperpłaszczyzn norm trójkątnych postaci (4.167) i (4.168)

Przykład 4.24 (Normy trójkątne algebraiczne)

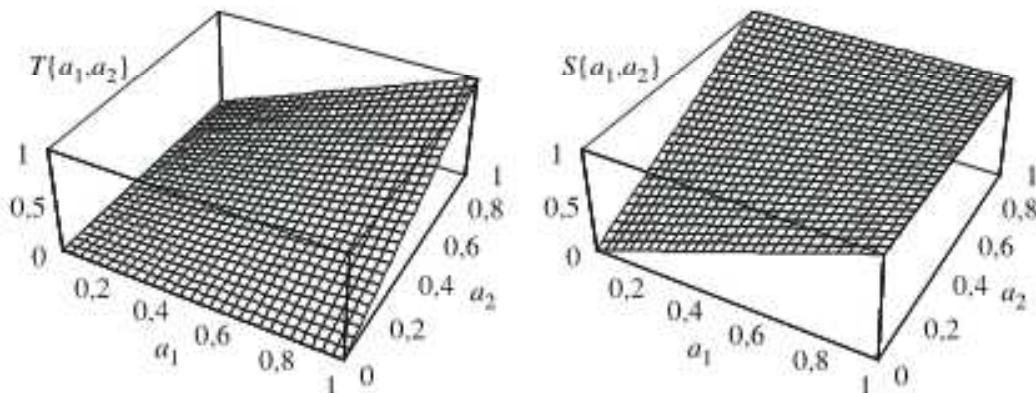
Normy trójkątne algebraiczne są opisane poniższymi zależnościami:

$$T_P\{a_1, a_2\} = a_1 a_2, \quad (4.171)$$

$$S_P\{a_1, a_2\} = a_1 + a_2 - a_1 a_2, \quad (4.172)$$

$$T_P\{a_1, a_2, \dots, a_n\} = \prod_{i=1}^n a_i, \quad (4.173)$$

$$S_P\{a_1, a_2, \dots, a_n\} = 1 - \prod_{i=1}^n (1 - a_i). \quad (4.174)$$



Rys. 4.30. Kształt hiperpłaszczyzn norm trójkątnych postaci (4.171) i (4.172)

Normy trójkątne algebraiczne są dualnymi normami trójkątnymi typu *strict*.

Przykład 4.25 (Normy trójkątne Łukasiewicza)

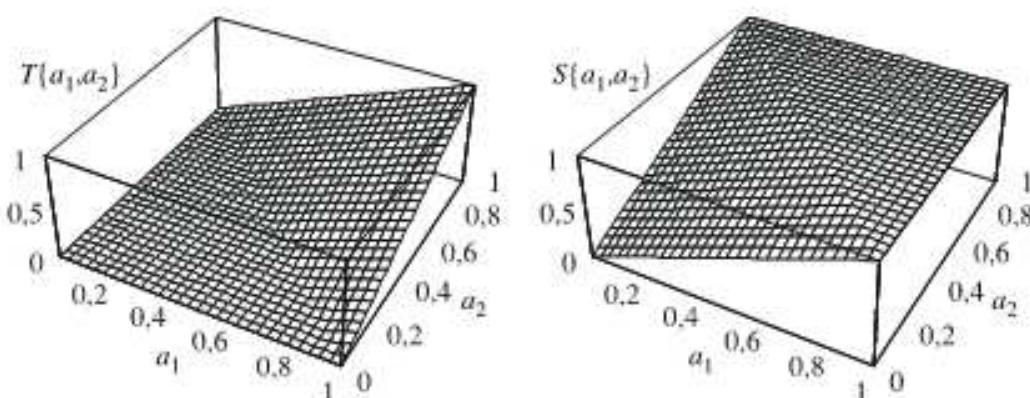
Normy trójkątne Łukasiewicza są opisane poniższymi zależnościami:

$$T_L\{a_1, a_2\} = \max\{a_1 + a_2 - 1, 0\}, \quad (4.175)$$

$$S_L\{a_1, a_2\} = \min\{a_1 + a_2, 1\}, \quad (4.176)$$

$$T_L\{a_1, a_2, \dots, a_n\} = \max \left\{ \sum_{i=1}^n a_i - (n-1), 0 \right\}, \quad (4.177)$$

$$S_L\{a_1, a_2, \dots, a_n\} = \min \left\{ \sum_{i=1}^n a_i, 1 \right\}. \quad (4.178)$$



Rys. 4.31. Kształt hiperplaszczyzn norm trójkątnych postaci (4.175) i (4.176)

Normy trójkątne Łukasiewicza są dualnymi normami trójkątnymi typu *nilpotent*.

Przykład 4.26 (Normy trójkątne graniczne)

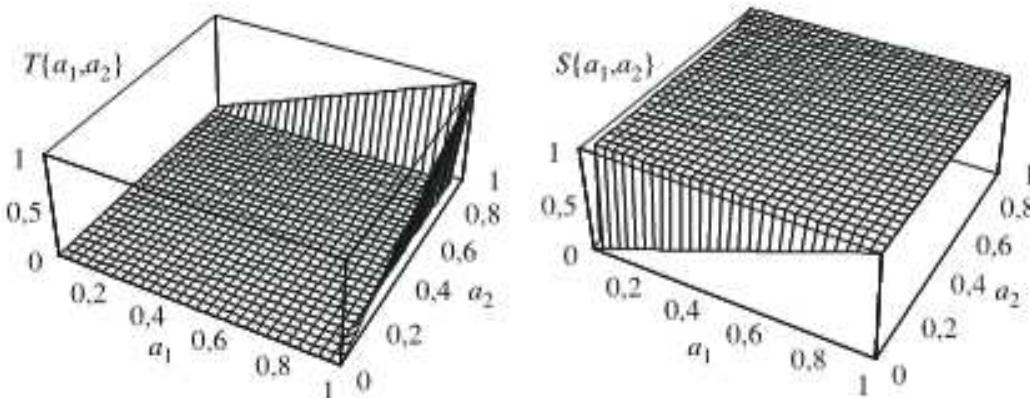
Normy trójkątne graniczne są opisane poniższymi zależnościami.

$$T_D\{a_1, a_2\} = \begin{cases} 0, & \text{jeżeli } S_M\{a_1, a_2\} < 1, \\ T_M\{a_1, a_2\}, & \text{jeżeli } S_M\{a_1, a_2\} = 1. \end{cases} \quad (4.179)$$

$$S_D\{a_1, a_2\} = \begin{cases} 1, & \text{jeżeli } T_M\{a_1, a_2\} > 0, \\ S_M\{a_1, a_2\}, & \text{jeżeli } T_M\{a_1, a_2\} = 0. \end{cases} \quad (4.180)$$

$$T_D\{a_1, a_2, \dots, a_n\} = \begin{cases} 0, & \text{jeżeli } S_M\{a_1, a_2, \dots, a_n\} < 1, \\ T_M\{a_1, a_2, \dots, a_n\}, & \text{jeżeli } S_M\{a_1, a_2, \dots, a_n\} = 1. \end{cases} \quad (4.181)$$

$$S_D\{a_1, a_2, \dots, a_n\} = \begin{cases} 1, & \text{jeżeli } T_M\{a_1, a_2, \dots, a_n\} > 0, \\ S_M\{a_1, a_2, \dots, a_n\}, & \text{jeżeli } T_M\{a_1, a_2, \dots, a_n\} = 0. \end{cases} \quad (4.182)$$



Rys. 4.32. Kształt hiperplaszczyzn norm trójkątnych postaci (4.179) i (4.180)

Warto wspomnieć, że wszystkie normy trójkątne spełniają następujące nierówności:

$$S_M\{a_1, a_2, \dots, a_n\} \leq S\{a_1, a_2, \dots, a_n\} \leq S_D\{a_1, a_2, \dots, a_n\}, \quad (4.183)$$

$$T_D\{a_1, a_2, \dots, a_n\} \leq T\{a_1, a_2, \dots, a_n\} \leq T_M\{a_1, a_2, \dots, a_n\}. \quad (4.184)$$

Z normami trójkątnymi bezpośrednio związane są generatorzy norm trójkątnych. Funkcje zwane generatorami norm trójkątnych pozwalają określić właściwości norm

trójkątnych, np. rozróżnić normy trójkątne typu *nilpotent* lub *strict*, umożliwiają uogólnienie dwuargumentowych norm trójkątnych na normy trójkątne wieloargumentowe, a także pozwalają wyprowadzać nowe normy trójkątne.

DEFINICJA 4.30

Funkcję postaci

$$t_{\text{mul}} : [0, 1] \rightarrow [t_{\text{mul}}(0), 1] \quad (4.185)$$

nazywamy *multiplikatywnym generatorem archimedesowej t-normy*

$$T\{a_1, a_2, \dots, a_n\} = t_{\text{mul}}^{-1}\left(\left(\prod_{i=1}^n t_{\text{mul}}(a_i)\right) \vee t_{\text{mul}}(0)\right) \quad (4.186)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$. Dla archimedesowej t-normy typu st (*strict*) mamy

$$t_{\text{mul}}(0) = 0. \quad (4.187)$$

Zatem wzór (4.186) przybiera postać

$$T\{a_1, a_2, \dots, a_n\} = t_{\text{mul}}^{-1}\left(\prod_{i=1}^n t_{\text{mul}}(a_i)\right) \quad (4.188)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

DEFINICJA 4.31

Funkcję postaci

$$t_{\text{add}} : [0, 1] \rightarrow [0, t_{\text{add}}(0)] \quad (4.189)$$

nazywamy *addytywnym generatorem archimedesowej t-normy*

$$T\{a_1, a_2, \dots, a_n\} = t_{\text{add}}^{-1}\left(\left(\sum_{i=1}^n t_{\text{add}}(a_i)\right) \wedge t_{\text{add}}(0)\right) \quad (4.190)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

Dla archimedesowej t-normy typu st (*strict*) mamy

$$t_{\text{add}}(0) = \infty. \quad (4.191)$$

Wówczas wzór (4.190) przybiera postać

$$T\{a_1, a_2, \dots, a_n\} = t_{\text{add}}^{-1}\left(\sum_{i=1}^n t_{\text{add}}(a_i)\right) \quad (4.192)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

DEFINICJA 4.32

Funkcję postaci

$$s_{\text{mul}} : [0, 1] \rightarrow [s_{\text{mul}}(1), 1] \quad (4.193)$$

nazywamy *multiplikatywnym generatorem archimedesowej t-konormy*

$$S\{a_1, a_2, \dots, a_n\} = s_{\text{mul}}^{-1}\left(\left(\prod_{i=1}^n s_{\text{mul}}(a_i)\right) \vee s_{\text{mul}}(1)\right) \quad (4.194)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

Dla archimedesowej t -konormy typu st (*strict*) mamy

$$s_{\text{mul}}(1) = 0. \quad (4.195)$$

Zatem wzór (4.194) przybiera postać

$$S\{a_1, a_2, \dots, a_n\} = s_{\text{mul}}^{-1}\left(\prod_{i=1}^n s_{\text{mul}}(a_i)\right) \quad (4.196)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

DEFINICJA 4.33

Funkcję postaci

$$s_{\text{add}} : [0, 1] \rightarrow [0, s_{\text{add}}(1)] \quad (4.197)$$

nazywamy *addytywnym generatorem archimedesowej t -konormy (nilpotent)*

$$S\{a_1, a_2, \dots, a_n\} = s_{\text{add}}^{-1}\left(\left(\sum_{i=1}^n s_{\text{add}}(a_i)\right) \wedge s_{\text{add}}(1)\right) \quad (4.198)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

Dla archimedesowej t -konormy typu st (*strict*) mamy

$$s_{\text{add}}(1) = \infty. \quad (4.199)$$

Zatem wzór (4.198) przybiera postać

$$S\{a_1, a_2, \dots, a_n\} = s_{\text{add}}^{-1}\left(\sum_{i=1}^n s_{\text{add}}(a_i)\right) \quad (4.200)$$

dla $a_1, a_2, \dots, a_n \in [0, 1]$.

Powiązanie generatora multiplikatywnego i addytywnego archimedesowej t -normy jest następujące:

$$t_{\text{mul}}(a) = \exp(-t_{\text{add}}(a)), \quad (4.201)$$

$$t_{\text{add}}(a) = -\ln(t_{\text{mul}}(a)). \quad (4.202)$$

Powiązanie generatorów multiplikatywnych i addytywnych dualnych norm trójkątnych jest następujące:

$$s_{\text{mul}}(a) = t_{\text{mul}}(1-a), \quad (4.203)$$

$$s_{\text{add}}(a) = t_{\text{add}}(1-a). \quad (4.204)$$

Uwaga 4.7

Niech f będzie dowolnym multiplikatywnym generatorem archimedesowej t -normy T . Wówczas t -norma T jest typu *nilpotent* wtedy i tylko wtedy, gdy $f(0) > 0$, oraz jest typu *strict* wtedy i tylko wtedy, gdy $f(0) = 0$. Wynik ten podano w monografii [144]. Jeżeli natomiast f jest dowolnym addytywnym generatorem archimedesowej t -normy T , to t -norma T jest typu *nilpotent* wtedy i tylko wtedy, gdy $f(0) < \infty$, oraz jest typu *strict* wtedy i tylko wtedy, gdy $f(0) = \infty$. Rezultat ten podano w monografii [111].

Przykład 4.27

Rozważmy multiplikatywny generator archimedesowej t -normy dany wzorem

$$t_{\text{mul}}(a) = a^p, \quad p > 0. \quad (4.205)$$

Korzystając z zależności (4.201)–(4.204), można wyznaczyć addytywny generator archimedesowej t -normy oraz multiplikatywny i addytywny generator archimedesowej t -konormy w sposób następujący:

$$t_{\text{add}}(a) = -\ln(t_{\text{mul}}(a)) = -p \ln(a), \quad (4.206)$$

$$s_{\text{mul}}(a) = t_{\text{mul}}(1-a) = (1-a)^p, \quad (4.207)$$

$$s_{\text{add}}(a) = t_{\text{add}}(1-a) = -p \ln(1-a). \quad (4.208)$$

Chcąc wykorzystać wyżej wymienione generatory do wygenerowania norm trójkątnych, należy określić ich funkcje odwrotne

$$t_{\text{mul}}^{-1}(a) = a^{\frac{1}{p}}, \quad (4.209)$$

$$t_{\text{add}}^{-1}(a) = \exp(-a)^{\frac{1}{p}}, \quad (4.210)$$

$$s_{\text{mul}}^{-1}(a) = 1 - a^{\frac{1}{p}}, \quad (4.211)$$

$$s_{\text{add}}^{-1}(a) = 1 - \exp(-a)^{\frac{1}{p}}. \quad (4.212)$$

Zauważmy, że dla generatora danego wzorem (4.205) mamy

$$t_{\text{mul}}(0) = s_{\text{mul}}(1) = 0. \quad (4.213)$$

Warunki brzegowe wskazują, że wykorzystane generatory są związane z normami trójkątnymi typu *strict*. Dlatego do wyznaczenia tych norm wykorzystamy zależności (4.188), (4.192), (4.196) lub (4.200). Zatem

$$\begin{aligned} T\{a_1, a_2\} &= t_{\text{mul}}^{-1}(t_{\text{mul}}(a_1)t_{\text{mul}}(a_2)) \\ &= t_{\text{add}}^{-1}(t_{\text{add}}(a_1) + t_{\text{add}}(a_2)) \\ &= a_1 a_2 \\ &= T_P\{a_1, a_2\} \end{aligned} \quad (4.214)$$

oraz

$$\begin{aligned} S\{a_1, a_2\} &= s_{\text{mul}}^{-1}(s_{\text{mul}}(a_1)s_{\text{mul}}(a_2)) \\ &= s_{\text{add}}^{-1}(s_{\text{add}}(a_1) + s_{\text{add}}(a_2)) \\ &= a_1 + a_2 - a_1 a_2 \\ &= S_P\{a_1, a_2\}. \end{aligned} \quad (4.215)$$

Przykład 4.28

Rozważmy multiplikatywny generator archimedesowej t -normy

$$t_{\text{mul}}(a) = \exp(a-1). \quad (4.216)$$

Korzystając z zależności (4.201)–(4.204), możemy wyznaczyć addytywny generator archimedesowej t -normy oraz multiplikatywny i addytywny generator archimedesowej

t -konormy w sposób następujący:

$$t_{\text{add}}(a) = -\ln(t_{\text{mul}}(a)) = 1 - a, \quad (4.217)$$

$$s_{\text{mul}}(a) = t_{\text{mul}}(1 - a) = \exp(-a), \quad (4.218)$$

$$s_{\text{add}}(a) = t_{\text{add}}(1 - a) = a. \quad (4.219)$$

Chcąc wykorzystać wyżej wymienione generatory do wygenerowania norm trójkątnych, należy określić ich funkcje odwrotne

$$t_{\text{mul}}^{-1}(a) = 1 + \ln(a), \quad (4.220)$$

$$t_{\text{add}}^{-1}(a) = 1 - a, \quad (4.221)$$

$$s_{\text{mul}}^{-1}(a) = -\ln(a), \quad (4.222)$$

$$s_{\text{add}}^{-1}(a) = a. \quad (4.223)$$

Zauważmy, że dla generatora danego wzorem (4.216) mamy

$$t_{\text{mul}}(0) = s_{\text{mul}}(1) = \frac{1}{e}. \quad (4.224)$$

Warunki brzegowe wskazują, że wykorzystane generatory są związane z normami trójkątnymi typu *nilpotent*. Dlatego do wyznaczenia tych norm wykorzystamy zależności (4.186), (4.190), (4.194) lub (4.198). Wówczas

$$\begin{aligned} T\{a_1, a_2\} &= t_{\text{mul}}^{-1}((t_{\text{mul}}(a_1)t_{\text{mul}}(a_2)) \vee t_{\text{mul}}(0)) \\ &= t_{\text{add}}^{-1}((t_{\text{add}}(a_1) + t_{\text{add}}(a_2)) \wedge t_{\text{add}}(0)) \\ &= \max\{a_1 + a_2 - 1, 0\} \\ &= T_L\{a_1, a_2\} \end{aligned} \quad (4.225)$$

oraz

$$\begin{aligned} S\{a_1, a_2\} &= s_{\text{mul}}^{-1}((s_{\text{mul}}(a_1)s_{\text{mul}}(a_2)) \vee s_{\text{mul}}(1)) \\ &= s_{\text{add}}^{-1}((s_{\text{add}}(a_1) + s_{\text{add}}(a_2)) \wedge s_{\text{add}}(1)) \\ &= \min\{a_1 + a_2, 1\} \\ &= S_L\{a_1, a_2\}. \end{aligned} \quad (4.226)$$

W podrozdziale 4.3 zdefiniowaliśmy podstawowe operacje na zbiorach rozmytych (przecięcie, suma i iloczyn kartezjański), korzystając z operatorów typu min/max. Obecnie uogólnimy te definicje, stosując dowolne normy trójkątne.

DEFINICJA 4.34

Przecięciem n zbiorów rozmytych $A_1, A_2, \dots, A_n \subseteq \mathbf{X}$ jest zbiór rozmyty $A = A_1 \cap A_2 \cap \dots \cap A_n = \bigcap_{i=1}^n A_i$ określony przez funkcję przynależności

$$\mu_A(\mathbf{x}) = \prod_{i=1}^n \mu_{A_i}(\mathbf{x}). \quad (4.227)$$

DEFINICJA 4.35

Sumą zbiorów rozmytych $A_1, A_2, \dots, A_n \subseteq \mathbf{X}$ jest zbiór rozmyty $A = A_1 \cup A_2 \cup \dots \cup A_n =$

$\bigcup_{i=1}^n A_i$ określony przez funkcję przynależności

$$\mu_A(\mathbf{x}) = \bigwedge_{i=1}^n \mu_{A_i}(\mathbf{x}). \quad (4.228)$$

DEFINICJA 4.36

Iloczyn kartezjański n zbiorów rozmytych $A_1 \subseteq \mathbf{X}_1, A_2 \subseteq \mathbf{X}_2, \dots, A_n \subseteq \mathbf{X}_n$, oznaczany $A_1 \times A_2 \times \dots \times A_n$, jest określony przez

$$\begin{aligned} \mu_{A_1 \times A_2 \times \dots \times A_n}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) &= T\{\mu_{A_1}(\mathbf{x}_1), \mu_{A_2}(\mathbf{x}_2), \dots, \mu_{A_n}(\mathbf{x}_n)\} \\ &= \bigwedge_{i=1}^n \{\mu_{A_i}(\mathbf{x}_i)\}. \end{aligned} \quad (4.229)$$

Negacje są rozszerzeniami logicznego zaprzeczenia. Pełnią funkcję operatorów zaprzeczenia w rozmytych modelach lingwistycznych.

DEFINICJA 4.37

- (i) Nierosnącą funkcję $N : [0, 1] \rightarrow [0, 1]$ nazywamy *negacją q*, jeżeli $N(0) = 1$ i $N(1) = 0$.
- (ii) Negację $N : [0, 1] \rightarrow [0, 1]$ nazywamy negacją *typu st (strict)*, jeżeli jest ciągła i malejąca.
- (iii) Negację typu st (*strict*) nazywamy *typu strong*, jeżeli jest inwolucją, tzn. $N(N(a)) = a$.

Poniżej zamieszczono wykaz wybranych negacji.

Przykład 4.29

Negacja Zadeha jest opisana poniższą zależnością

$$N(a) = 1 - a. \quad (4.230)$$

Łatwo zauważyc, że negacja Zadeha jest negacją typu *strong*.

Przykład 4.30

Negacja Yagera jest opisana zależnością

$$N(a) = (1 - a^p)^{\frac{1}{p}}, \quad p > 0. \quad (4.231)$$

Negacja Yagera jest negacją typu *strict*.

Przykład 4.31

Negacja Sugeno jest opisana poniższą zależnością

$$N(a) = \frac{1 - a}{1 + pa}, \quad p > -1. \quad (4.232)$$

Negacja Sugeno jest negacją typu *strong*.

Uwaga 4.8

Jeżeli t -norma, t -konorma oraz negacja N typu *strong* spełniają zależności

$$\bigwedge_{i=1}^n \{a_i\} = N^{-1} \left(\bigwedge_{i=1}^n \{N(a_i)\} \right) \quad (4.233)$$

oraz

$$\bigcap_{i=1}^n \{a_i\} = N^{-1} \left(\bigcap_{i=1}^n \{N(a_i)\} \right), \quad (4.234)$$

to T , S i N stanowią tzw. trójkę De Morgana i mówimy, że t -norma T oraz t -konorma S są N -dualne względem siebie. Warto zauważyć, że normy trójkątne w definicji 4.26 są dualne przy zastosowaniu negacji Zadeha $N(a) = 1 - a$.

4.7. Relacje rozmyte i ich właściwości

Jednym z podstawowych pojęć teorii zbiorów rozmytych jest pojęcie relacji rozmytej. Relacje rozmyte pozwalają sformalizować nieprecyzyjne sformułowania typu „ x jest prawie równe y ” lub „ x jest znacznie większe od y ”. Podamy definicje relacji rozmytej oraz złożień relacji rozmytych. Szczególnie ważna z punktu widzenia zastosowań jest definicja 4.39 dotycząca złożenia zbioru rozmytego i relacji rozmytej.

DEFINICJA 4.38

Relacją rozmytą R między dwoma niepustymi zbiorami (nierożmytymi) \mathbf{X} i \mathbf{Y} nazywamy zbiór rozmyty określony na iloczynie kartezjańskim $\mathbf{X} \times \mathbf{Y}$, tzn.

$$R \subseteq \mathbf{X} \times \mathbf{Y} = \{(x, y) : x \in \mathbf{X}, y \in \mathbf{Y}\}. \quad (4.235)$$

Innymi słowy, relacja rozmyta jest zbiorem par

$$R = \{(x, y), \mu_R(x, y)\}, \quad \forall_{x \in \mathbf{X}} \forall_{y \in \mathbf{Y}}, \quad (4.236)$$

gdzie

$$\mu_R : \mathbf{X} \times \mathbf{Y} \rightarrow [0, 1] \quad (4.237)$$

jest funkcją przynależności. Funkcja ta każdej parze (x, y) , $x \in \mathbf{X}$, $y \in \mathbf{Y}$ przypisuje jej stopień przynależności $\mu_R(x, y)$, który interpretuje się jako siłę powiązania między elementami $x \in \mathbf{X}$ i $y \in \mathbf{Y}$. Zgodnie z przyjętą przez nas konwencją (podrozdz. 4.2) relację rozmytą można zapisać w postaci

$$R = \sum_{\mathbf{X} \times \mathbf{Y}} \frac{\mu_R(x, y)}{(x, y)} \quad (4.238)$$

lub

$$R = \int_{\mathbf{X} \times \mathbf{Y}} \frac{\mu_R(x, y)}{(x, y)}. \quad (4.239)$$

Przykład 4.32

Zastosujemy definicję 4.38 do sformalizowania nieprecyzyjnego stwierdzenia „ y jest mniej więcej równe x ”. Niech $\mathbf{X} = \{3, 4, 5\}$ oraz $\mathbf{Y} = \{4, 5, 6\}$. Relację R możemy zdefiniować następująco:

$$R = \frac{1}{(4, 4)} + \frac{1}{(5, 5)} + \frac{0.8}{(3, 4)} + \frac{0.8}{(4, 5)} + \frac{0.8}{(5, 4)} + \frac{0.8}{(5, 6)} + \frac{0.6}{(3, 5)} + \frac{0.6}{(4, 6)} + \frac{0.4}{(3, 6)}. \quad (4.240)$$

Zatem funkcja przynależności $\mu_R(x, y)$ relacji R jest postaci

$$\mu_R(x, y) = \begin{cases} 1, & \text{jeżeli } x = y, \\ 0.8, & \text{jeżeli } |x - y| = 1, \\ 0.6, & \text{jeżeli } |x - y| = 2, \\ 0.4, & \text{jeżeli } |x - y| = 3. \end{cases} \quad (4.241)$$

Relację R można również zapisać za pomocą macierzy

$$\begin{matrix} & y_1 & y_2 & y_3 \\ x_1 & 0.8 & 0.6 & 0.4 \\ x_2 & 1 & 0.8 & 0.6 \\ x_3 & 0.8 & 1 & 0.8 \end{matrix} \quad (4.242)$$

w której $x_1 = 3, x_2 = 4, x_3 = 5$ oraz $y_1 = 4, y_2 = 5, y_3 = 6$.

Przykład 4.33

Niech $\mathbf{X} = \mathbf{Y} = [0, 120]$ będzie długością życia człowieka. Wówczas relacja R o funkcji przynależności

$$\mu_R(x, y) = \begin{cases} 0, & \text{jeżeli } x - y \leq 0, \\ \frac{x - y}{30}, & \text{jeżeli } 0 < x - y < 30, \\ 1, & \text{jeżeli } x - y \geq 30 \end{cases} \quad (4.243)$$

reprezentuje nieprecyzyjne stwierdzenie „osoba w wieku x jest dużo starsza od osoby w wieku y ”. Należy podkreślić, że relacja rozmyta R jest zbiorem rozmytym, a zatem pozostają w mocy podane w podrozdziale 4.3 definicje przecięcia, sumy i dopełnienia

$$\mu_{R \cap S}(x, y) = \min(\mu_R(x, y), \mu_S(x, y)), \quad (4.244)$$

$$\mu_{R \cup S}(x, y) = \max(\mu_R(x, y), \mu_S(x, y)), \quad (4.245)$$

$$\mu_{\bar{R}}(x, y) = 1 - \mu_R(x, y). \quad (4.246)$$

W teorii zbiorów rozmytych ważną rolę odgrywa pojęcie złożenia dwóch relacji rozmytych. Rozważmy trzy zbiory nierożmyte $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ oraz dwie relacje rozmyte $R \subseteq \mathbf{X} \times \mathbf{Y}$ i $S \subseteq \mathbf{Y} \times \mathbf{Z}$ o funkcjach przynależności $\mu_R(x, y)$ i $\mu_S(y, z)$.

DEFINICJA 4.39

Złożeniem typu sup- T relacji rozmytych $R \subseteq \mathbf{X} \times \mathbf{Y}$ i $S \subseteq \mathbf{Y} \times \mathbf{Z}$ nazywamy relację rozmytą $R \circ S \subseteq \mathbf{X} \times \mathbf{Z}$ o funkcji przynależności

$$\mu_{R \circ S}(x, z) = \sup_{y \in \mathbf{Y}} \{\mu_R(x, y) * \mu_S(y, z)\}. \quad (4.247)$$

Konkretna postać funkcji przynależności $\mu_{R \circ S}(x, z)$ złożenia $R \circ S$ zależy od przyjętej t -normy we wzorze (4.247). Jeżeli jako t -normę przyjmiemy \min , tzn. $T(a, b) = \min(a, b)$, to równość (4.247) można zapisać następująco:

$$\mu_{R \circ S}(x, z) = \sup_{y \in \mathbf{Y}} \{\min[\mu_R(x, y), \mu_S(y, z)]\}. \quad (4.248)$$

Wzór (4.248) znany jest w literaturze pod nazwą złożenia typu sup-min. Jeżeli zbiór \mathbf{Y} ma skończoną liczbę elementów, to złożenie typu sup-min sprowadza się do złożenia typu max-min postaci

$$\mu_{R \circ S}(x, z) = \max_{y \in \mathbf{Y}} \{\min[\mu_R(x, y), \mu_S(y, z)]\}. \quad (4.249)$$

Przykład 4.34

Założymy, że relacje R i S są reprezentowane przez macierze

$$R = \begin{bmatrix} 0.2 & 0.5 \\ 0.6 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} 0.3 & 0.6 & 0.8 \\ 0.7 & 0.9 & 0.4 \end{bmatrix}, \quad (4.250)$$

przy czym $\mathbf{X} = \{x_1, x_2\}$, $\mathbf{Y} = \{y_1, y_2\}$, $\mathbf{Z} = \{z_1, z_2, z_3\}$. Złożenie typu max-min relacji R i S ma postać

$$Q = R \circ S = \begin{bmatrix} 0.2 & 0.5 \\ 0.6 & 1 \end{bmatrix} \circ \begin{bmatrix} 0.3 & 0.6 & 0.8 \\ 0.7 & 0.9 & 0.4 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix}, \quad (4.251)$$

gdzie

$$q_{11} = \max[\min(0.2; 0.3), \min(0.5; 0.7)] = 0.5,$$

$$q_{12} = \max[\min(0.2; 0.6), \min(0.5; 0.9)] = 0.5,$$

$$q_{13} = \max[\min(0.2; 0.8), \min(0.5; 0.4)] = 0.4,$$

$$q_{21} = \max[\min(0.6; 0.3), \min(1; 0.7)] = 0.7,$$

$$q_{22} = \max[\min(0.6; 0.6), \min(1; 0.9)] = 0.9,$$

$$q_{23} = \max[\min(0.6; 0.8), \min(1; 0.4)] = 0.6.$$

Zatem

$$Q = \begin{bmatrix} 0.5 & 0.5 & 0.4 \\ 0.7 & 0.9 & 0.6 \end{bmatrix}. \quad (4.252)$$

W tabeli 4.1 zestawiono podstawowe właściwości relacji rozmytych, przy czym I oznacza macierz jednostkową, a O macierz zerową.

Tabela 4.1. Podstawowe właściwości relacji rozmytych

1	$R \circ I = I \circ R = R$
2	$R \circ O = O \circ R = O$
3	$(R \circ S) \circ T = R \circ (S \circ T)$
4	$R^m \circ R^n = R^{m+n}$
5	$(R^m)^n = R^{mn}$
6	$R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$
7	$R \circ (S \cap T) \subseteq (R \circ S) \cap (R \circ T)$
8	$S \subset T \rightarrow R \circ S \subset R \circ T$

Jak wspominaliśmy na wstępie podrozdziału 4.6, szczególnie ważne w różnych zastosowaniach jest złożenie zbioru rozmytego z relacją rozmytą. Złożenie tego typu będzie wielokrotnie stosowane w dalszej części rozdziału. Rozważmy zbiór rozmyty $A \subseteq \mathbf{X}$ oraz relację rozmytą $R \subseteq \mathbf{X} \times \mathbf{Y}$ o funkcjach przynależności $\mu_A(x)$ i $\mu_R(x, y)$.

DEFINICJA 4.40

Złożenie zbioru rozmytego $A \subseteq \mathbf{X}$ i relacji rozmytej $R \subseteq \mathbf{X} \times \mathbf{Y}$ oznaczamy $A \circ R$ i definiujemy jako zbiór rozmyty $B \subseteq \mathbf{Y}$

$$B = A \circ R \quad (4.253)$$

o funkcji przynależności

$$\mu_B(y) = \sup_{x \in \mathbf{X}} \{\mu_A(x) * \mu_R(x, y)\}. \quad (4.254)$$

Konkretna postać wzoru (4.254) zależy od przyjętej t -normy (patrz tab. 4.1) oraz od właściwości zbioru \mathbf{X} . Poniżej wyróżnimy 4 przypadki:

1) Jeżeli $T(a, b) = \min(a, b)$, to otrzymujemy złożenie typu sup-min

$$\mu_B(y) = \sup_{x \in \mathbf{X}} \{\min[\mu_A(x), \mu_R(x, y)]\}. \quad (4.255)$$

2) Jeżeli $T(a, b) = \min(a, b)$ oraz \mathbf{X} jest zbiorem o skończonej liczbie elementów, to otrzymujemy złożenie typu max-min

$$\mu_B(y) = \max_{x \in \mathbf{X}} \{\min[\mu_A(x), \mu_R(x, y)]\}. \quad (4.256)$$

3) Jeżeli $T(a, b) = a \cdot b$, to otrzymujemy złożenie typu sup-iloczyn (ang. *sup-product*)

$$\mu_B(y) = \sup_{x \in \mathbf{X}} \{\mu_A(x) \cdot \mu_R(x, y)\}. \quad (4.257)$$

4) Jeżeli $T(a, b) = a \cdot b$ oraz \mathbf{X} jest zbiorem o skończonej liczbie elementów, to otrzymujemy złożenie typu max-iloczyn (ang. *max-product*)

$$\mu_B(y) = \max_{x \in \mathbf{X}} \{\mu_A(x) \cdot \mu_R(x, y)\}. \quad (4.258)$$

Przykład 4.35

Założmy, że $\mathbf{X} = \{x_1, x_2, x_3\}$ oraz $\mathbf{Y} = \{y_1, y_2\}$. Zbiór rozmyty A jest postaci

$$A = \frac{0.4}{x_1} + \frac{1}{x_2} + \frac{0.6}{x_3}, \quad (4.259)$$

natomiast relację R reprezentuje macierz

$$R = \begin{matrix} & y_1 & y_2 \\ x_1 & \left[\begin{matrix} 0.5 & 0.7 \\ 0.2 & 1 \end{matrix} \right] \\ x_2 & \\ x_3 & \end{matrix}. \quad (4.260)$$

Złożenie $A \circ R$ typu max-min wyznaczamy zgodnie ze wzorem (4.256). Rezultatem złożenia jest zbiór rozmyty B postaci

$$B = \frac{\mu_B(y_1)}{y_1} + \frac{\mu_B(y_2)}{y_2}, \quad (4.261)$$

przy czym

$$\mu_B(y_1) = \max\{\min(0.4; 0.5), \min(1; 0.2), \min(0.6; 0.9)\} = 0.6, \quad (4.262)$$

$$\mu_B(y_2) = \max\{\min(0.4; 0.7), \min(1; 1), \min(0.6; 0.3)\} = 1. \quad (4.263)$$

Zatem

$$B = \frac{0.6}{y_1} + \frac{1}{y_2}. \quad (4.264)$$

Złożenie $A \circ R$ można również zdefiniować jako projekcję przecięcia rozszerzenia cylindrycznego $ce(A)$ zbioru rozmytego A i relacji rozmytej R , na przestrzeń \mathbf{Y} , tzn.

$$B = A \circ R = \text{proj}\{ce(A) \cap R\} \text{ na } \mathbf{Y}, \quad (4.265)$$

gdzie operacje rozszerzenia cylindrycznego i projekcji zdefiniowano poniżej.

DEFINICJA 4.41

Rozszerzenie cylindryczne zbioru $A \subseteq \mathbf{X}$ w zbiór $ce(A) \subseteq \mathbf{X} \times \mathbf{Y}$ definiuje się następująco:

$$ce(A) = \int_{\mathbf{X} \times \mathbf{Y}} \frac{\mu_A(x)}{(x, y)}. \quad (4.266)$$

DEFINICJA 4.42

Projekcją zbioru rozmytego $A \subseteq \mathbf{X} \times \mathbf{Y}$ na przestrzeń \mathbf{Y} nazywamy operację określoną następująco:

$$\text{proj } A \text{ na } \mathbf{Y} = \int_{y \in \mathbf{Y}} \frac{\sup_{x \in \mathbf{X}} \mu_A(x, y)}{y}. \quad (4.267)$$

4.8. Przybliżone wnioskowanie

4.8.1. Podstawowe reguły wnioskowania w logice dwuwartościowej

W logice tradycyjnej (dwuwartościowej) wnioskujemy o prawdziwości pewnych zdań na podstawie prawdziwości innych zdań. Wnioskowanie to notujemy w postaci schematu: nad kreską poziomą piszemy wszystkie te zdania, na podstawie których wnioskujemy, pod kreską piszemy otrzymany wniosek. Schemat poprawnego wnioskowania ma tę właściwość, że jeżeli prawdziwe są wszystkie zdania nad kreską poziomą, to prawdziwe jest też zdanie pod kreską, gdyż ze zdań prawdziwych wynikać może tylko prawdziwy wniosek. W tym punkcie duże litery A i B symbolizują zdania, a nie zbiory rozmyte. Niech A oraz B będą zdaniami, przy czym zapis $A = 1$ ($B = 1$) oznacza, że wartością logiczną zdania $A(B)$ jest prawda, natomiast zapis $A = 0$ ($B = 0$) oznacza, że wartością logiczną zdania $A(B)$ jest fałsz. Przedstawimy teraz dwie reguły wnioskowania stosowane w logice dwuwartościowej.

DEFINICJA 4.43

Regułę wnioskowania *modus ponens* określa następujący schemat wnioskowania:

Przesłanka	A
Implikacja	$A \rightarrow B$
Wniosek	B

(4.268)

Przykład 4.36

Niech zdanie A ma postać „Jan jest kierowcą”, a zdanie B — „Jan ma prawo jazdy”. Zgodnie z regułą *modus ponens*, jeżeli $A = 1$, to również $B = 1$, gdyż jeżeli prawdą jest, że „Jan jest kierowcą”, to prawdą jest również, że „Jan ma prawo jazdy”. Innymi słowy, z prawdziwości przesłanki i implikacji (zdania nad kreską) wynika prawdziwość wniosku (zdanie pod kreską).

DEFINICJA 4.44

Regułę wnioskowania *modus tollens* określa następujący schemat wnioskowania:

Przesłanka	\bar{B}
Implikacja	$A \rightarrow B$
Wniosek	\bar{A}

(4.269)

Przykład 4.37

Kontynuując przykład 4.36, rozumiemy, że jeżeli „Jan nie ma prawa jazdy”, czyli $B = 0$ ($\bar{B} = 1$), to „Jan nie jest kierowcą”, czyli $A = 0$ ($\bar{A} = 1$). Również w tym przykładzie z prawdziwości przesłanki i implikacji wynika prawdziwość wniosku.

Przedstawiliśmy jedynie te dwie reguły wnioskowania w logice dwuwartościowej, które będą uogólnione na przypadek rozmyty. Oczywiście w logice dwuwartościowej znanych jest cały szereg innych reguł wnioskowania. Zainteresowanego czytelnika odsyłamy do bogatej literatury w tym zakresie (np. [234]).

4.8.2. Podstawowe reguły wnioskowania w logice rozmytej

Obecnie rozszerzymy podstawowe reguły wnioskowania w logice dwuwartościowej na przypadek rozmyty. Założymy, że występujące w regułach *modus ponens* (4.268) oraz *modus tollens* (4.269) zdania charakteryzują się pewnymi zbiorami rozmytymi. W ten sposób otrzymamy uogólnioną regułę wnioskowania *modus ponens* oraz uogólnioną regułę wnioskowania *modus tollens*.

4.8.2.1 Uogólniona rozmyta reguła wnioskowania *modus ponens***DEFINICJA 4.45**

Uogólnioną (rozmytą) regułę wnioskowania *modus ponens* określa następujący schemat wnioskowania:

Przesłanka	x jest A'
Implikacja	JEŻELI x jest A TO y jest B
Wniosek	y jest B'

(4.270)

gdzie $A, A' \subseteq X$ oraz $B, B' \subseteq Y$ są zbiorami rozmytymi, natomiast x i y są tzw. zmiennymi lingwistycznymi.

Występujące w powyższej definicji *zmienne lingwistyczne* to takie zmienne, które przyjmują jako swoje wartości słowa lub zdania wypowiadane w języku naturalnym. Jako przykłady można podać stwierdzenia typu „mała prędkość”, „umiarkowana temperatura” lub „młody człowiek”. Stwierdzenia te można sformalizować poprzez przyporządkowanie im pewnych zbiorów rozmytych. Należy podkreślić, że zmienne lingwistyczne oprócz wartości słownych mogą także przyjmować wartości liczbowe, jak zwykle zmienne matematyczne. Następny przykład ilustruje uogólnioną (rozmytą) regułę wnioskowania *modus ponens* oraz przybliża czytelnikowi pojęcie zmiennej lingwistycznej.

Przykład 4.38

Rozważmy następujący schemat wnioskowania

Przesłanka	Prędkość samochodu jest duża
Implikacja	Jeżeli prędkość samochodu jest bardzo duża to poziom hałasu jest wysoki
Wniosek	Poziom hałasu w samochodzie jest średnio-wysoki

(4.271)

W powyższym schemacie przesłanka, implikacja i wniosek są nieprecyzyjnymi stwierdzeniami. Jako zmienne lingwistyczne wyróżnimy: x — prędkość samochodu, y — poziom hałasu. Zbiór

$$T_1 = \{„mała”, „średnia”, „duża”, „bardzo duża”\}$$

jest zbiorem wartości zmiennej lingwistycznej x . Podobnie zbiór

$$T_2 = \{„mały”, „średni”, „średniowysoki”, „wysoki”\}$$

jest zbiorem wartości zmiennej lingwistycznej y .

Do każdego elementu zbioru T_1 i T_2 można przyporządkować odpowiedni zbiór rozmyty. Analizując schematy wnioskowania (4.270) oraz (4.271), otrzymujemy następujące zbiory rozmyte:

$$A = „bardzo duża prędkość samochodu”,$$

$$A' = „duża prędkość samochodu”,$$

oraz

$$B = „wysoki poziom hałasu”,$$

$$B' = „średniowysoki poziom hałasu”.$$

Czytelnik może zaproponować funkcje przynależności dla tych zbiorów rozmytych, podobnie jak to przedstawiono na rysunku 4.12. Omówimy różnicę między regułą nierożmytą (4.268) i rozmytą (4.270). W obu przypadkach implikacja jest tej samej postaci $A \rightarrow B$, gdzie A i B są zdaniem (reguła (4.268)) bądź zbiorami rozmytymi (reguła (4.270)). Jednakże zdanie A w implikacji reguły nierożmytej widnieje również w przesłance tej reguły. Natomiast przesłanka reguły rozmytej nie dotyczy zbioru rozmytego A , ale odnosi się do pewnego zbioru rozmytego A' , który może być w pewnym sensie zbliżony do A , lecz niekoniecznie $A = A'$. W przykładzie 4.38 zbiór rozmyty $A =$ „bardzo duża prędkość samochodu” nie jest równy zbiorowi rozmytemu $A' =$ „duża prędkość samochodu”. W rezultacie wnioski schematów (4.268) i (4.270) są różne. Wniosek reguły rozmytej odnosi się do pewnego zbioru rozmytego B' , który jest określony przez złożenie zbioru rozmytego A' i rozmytej implikacji $A \rightarrow B$, tzn.

$$B' = A' \circ (A \rightarrow B). \quad (4.272)$$

Rozmyta implikacja $A \rightarrow B$ jest równoważna pewnej relacji rozmytej $R \in \mathbf{X} \times \mathbf{Y}$ o funkcji przynależności $\mu_R(x, y)$. Zatem funkcję przynależności zbioru rozmytego B' można wyznaczyć za pomocą wzoru (4.254), który zapisujemy jako

$$\mu_{B'}(y) = \sup_{x \in \mathbf{X}} \{\mu_{A'}(x) * \mu_{A \rightarrow B}(x, y)\}, \quad (4.273)$$

przy czym $\mu_{A \rightarrow B}(x, y) = \mu_R(x, y)$. W szczególnym przypadku, gdy t -norma jest typu min, wzór (4.273) przybiera postać

$$\mu_{B'}(y) = \sup_{x \in \mathbf{X}} \{\min[\mu_{A'}(x), \mu_{A \rightarrow B}(x, y)]\}. \quad (4.274)$$

Czytelnik z łatwością zauważ, że jeżeli

$$A' = A, \quad \text{to } B' = B, \quad (4.275)$$

i uogólniona rozmyta reguła wnioskowania *modus ponens* (4.270) redukuje się do omówionej w punkcie 4.8.1 reguły *modus ponens* (4.268).

Założymy teraz, że zachodzi implikacja $A \rightarrow B$ w schemacie (4.270), natomiast zbiór rozmyty A' (przesłanka) jest równy kolejno:

- 1) $A' = A$,
- 2) $A' =$ „bardzo A ”, przy czym $\mu_{A'}(x) = \mu_A^2(x)$,
- 3) $A' =$ „mniej więcej A ”, przy czym $\mu_{A'}(x) = \mu_A^{1/2}(x)$,
- 4) $A' =$ „nie A ”, przy czym $\mu_{A'}(x) = 1 - \mu_A(x)$.

Zbiór rozmyty „bardzo A ” określa się przez operację koncentracji (4.75), zbiór rozmyty „mniej więcej A ” określa się przez operację rozcieńczenia (4.76), natomiast zbiór rozmyty „nie A ” określa się przez operację dopełnienia (4.59). W tabeli 4.2 pokazano (patrz [58]) oczywiste relacje, które mogą zachodzić między zbiorami rozmytymi A' i B' . Relacja 1 to schemat *modus ponens* (4.268), relacje 2b i 3b zachodzą, gdy nie ma ścisłego związku między A' i B' , relacja 4a oznacza, że z przesłanki x jest „nie A ” nie można wyciągnąć wniosku o y .

Tabela 4.2. Intuicyjne relacje między przesłankami i wnioskami uogólnionej rozmytej reguły *modus ponens*

Relacja	Przesłanka x jest A'	Wniosek y jest B'
1	x jest A	y jest B
2a	x jest „bardzo A ”	y jest „bardzo B ”
2b	x jest „bardzo A ”	y jest B
3a	x jest „mniej więcej A ”	y jest „mniej więcej B ”
3b	x jest „mniej więcej A ”	y jest B
4a	x jest „nie A ”	y jest nieokreślone
4b	x jest „nie A ”	y jest „nie B ”

4.8.2.2. Uogólniona rozmyta reguła wnioskowania *modus tollens*

DEFINICJA 4.46

Uogólnioną (rozmytą) regułę wnioskowania *modus tollens* określa następujący schemat wnioskowania:

Przesłanka Implikacja	y jest B' JEŻELI x jest A TO y jest B
Wniosek	x jest A'

(4.276)

gdzie $A, A' \subseteq \mathbf{X}$ oraz $B, B' \subseteq \mathbf{Y}$ są zbiorami rozmytymi, natomiast x i y są zmiennymi lingwistycznymi.

Przykład 4.39

Przykład ten nawiązuje do przykładu 4.38, przy czym pozostaje w mocy opis, który następuje po schemacie (4.271)

Przesłanka Implikacja	Poziom hałasu w samochodzie jest średniowysoki Jeżeli prędkość samochodu jest bardzo duża, to poziom hałasu jest wysoki
Wniosek	Prędkość samochodu jest duża

(4.277)

Zbiór rozmyty A' we wniosku schematu (4.276) jest określony przez złożenie relacji

$$A' = (A \rightarrow B) \circ B', \quad (4.278)$$

przy czym

$$\mu_{A'}(x) = \sup_{y \in \mathbf{Y}} \{\mu_{A \rightarrow B}(x, y) * \mu_{B'}(y)\}. \quad (4.279)$$

Jeżeli t -norma jest typu min, to wzór (4.279) przybiera postać

$$\mu_{A'}(x) = \sup_{y \in \mathbf{Y}} \{\min[\mu_{A \rightarrow B}(x, y), \mu_B(y)]\}. \quad (4.280)$$

Jeżeli

$$A' = \overline{A} \quad \text{oraz} \quad B' = \overline{B}, \quad (4.281)$$

to uogólniona rozmyta reguła wnioskowania *modus tollens* (4.276) redukuje się do omówionej w punkcie 4.8.1 reguły *modus tollens* (4.269). W tabeli 4.3 pokazano oczywiste relacje między przesłankami i wnioskami uogólnionej rozmytej reguły *modus tollens*.

Tabela 4.3. Intuicyjne relacje między przesłankami i wnioskami uogólnionej rozmytej reguły *modus tollens*

Relacja	Przesłanka y jest B'	Wniosek x jest A'
1	y jest „nie B ”	x jest „nie A ”
2	y jest „nie bardzo B ”	x jest „bardzo A ”
3	y jest „mniej więcej B ”	x jest „mniej więcej A ”
4a	y jest B	x jest nieokreślone
4b	y jest B	y jest A

4.8.3. Reguły wnioskowania dla modelu Mamdaniego

W poprzednim punkcie omówiliśmy uogólnione rozmyte schematy wnioskowania *modus ponens* oraz *modus tollens*. Funkcje przynależności (4.273) i (4.279) we wnioskach tych schematów zależą od funkcji przynależności $\mu_{A \rightarrow B}(x, y)$ rozmytej implikacji $A \rightarrow B$, która jest równoważna pewnej relacji rozmytej $R \subseteq \mathbf{X} \times \mathbf{Y}$. Przedstawimy różne sposoby wyznaczania funkcji $\mu_{A \rightarrow B}(x, y)$ na podstawie znajomości funkcji przynależności $\mu_A(x)$ i $\mu_B(y)$. W przypadku modelu Mamdaniego funkcje przynależności $\mu_{A \rightarrow B}(x, y)$ wyznaczamy w sposób następujący:

$$\mu_{A \rightarrow B}(x, y) = T(\mu_A(x), \mu_B(y)), \quad (4.282)$$

gdzie T jest dowolną t -normą. Najczęściej stosuje się regułę typu minimum określoną następująco:

- **Reguła typu minimum**

$$\mu_{A \rightarrow B}(x, y) = \mu_R(x, y) = \mu_A(x) \wedge \mu_B(y) = \min[\mu_A(x), \mu_B(y)]. \quad (4.283)$$

Inną znaną regułą jest reguła typu iloczyn (znana pod nazwą reguły Larsena):

- **Reguła typu iloczyn (Larsena)**

$$\mu_{A \rightarrow B}(x, y) = \mu_R(x, y) = \mu_A(x) \cdot \mu_B(y). \quad (4.284)$$

Tabela 4.4. Ilustracja działania reguł typu Mamdaniego

$\mu_A(x)$	$\mu_B(y)$	$\min[\mu_A(x), \mu_B(y)]$	$\mu_A(x)\mu_B(y)$	$\mu_{A \rightarrow B}(x, y)$
0	0	0	0	1
0	1	0	0	1
1	0	0	0	0
1	1	1	1	1

Należy podkreślić, że reguły typu Mamdaniego nie są implikacjami w sensie logicznym, co można łatwo wykazać, analizując tabelę 4.4.

Uwaga 4.9

Chociaż reguły Mamdaniego nie są implikacjami w sensie logicznym, to w literaturze często można spotkać błędne określenie tych reguł jako rozmytych implikacji. W monografii [134] Mendel reguły Mamdaniego nazywa „implikacjami inżynierskimi”, aby odróżnić je od implikacji rozmytych spełniających warunki definicji 4.47 przedstawionej w następnym punkcie.

4.8.4. Reguły wnioskowania dla modelu logicznego

Przedstawimy różne reguły wnioskowania dla modelu logicznego z wykorzystaniem definicji implikacji rozmytej [55].

DEFINICJA 4.47

Implikacja rozmyta jest funkcją $I : [0, 1]^2 \rightarrow [0, 1]$ spełniającą następujące warunki:

- jeżeli $a_1 \leq a_3$, to $I(a_1, a_2) \geq I(a_3, a_2)$ dla wszystkich $a_1, a_2, a_3 \in [0, 1]$,
- jeżeli $a_2 \leq a_3$, to $I(a_1, a_2) \leq I(a_1, a_3)$ dla wszystkich $a_1, a_2, a_3 \in [0, 1]$,
- $I(0, a_2) = 1$ dla wszystkich $a_2 \in [0, 1]$,
- $I(a_1, 1) = 1$ dla wszystkich $a_1 \in [0, 1]$,
- $I(1, 0) = 0$.

W tabeli 4.5 przedstawiono najczęściej stosowane implikacje rozmyte spełniające wszystkie lub niektóre wymagania definicji 4.47.

Niektóre z rozmytych implikacji w tabeli 4.5 należą do specjalnych grup implikacji:

- S-implikacje* zdefiniowane w sposób następujący:

$$I(a, b) = S\{1 - a, b\}.$$

Przykładami *S-implikacji* są implikacje 1, 2, 3 i 4 w tabeli 4.5. Spełniają one wszystkie warunki definicji 4.47.

- R-implikacje* zdefiniowane w sposób następujący:

$$I(a, b) = \sup_z \{z \mid T\{a, z\} \leq b\}, \quad a, b \in [0, 1].$$

Przykładami *R-implikacji* są implikacje 6 i 7 w tabeli 4.5. Spełniają one wszystkie warunki definicji 4.47.

Tabela 4.5. Implikacje rozmyte

Lp.	Nazwa	Implikacja
1	Kleene–Dienes (binarna)	$\max\{1 - a, b\}$
2	Łukasiewicz	$\min\{1, 1 - a + b\}$
3	Reichenbach	$1 - a + a \cdot b$
4	Fodor	$\begin{cases} 1, & \text{jeżeli } a \leq b \\ \max\{1 - a, b\}, & \text{jeżeli } a > b \end{cases}$
5	Rescher	$\begin{cases} 1, & \text{jeżeli } a \leq b \\ 0, & \text{jeżeli } a > b \end{cases}$
6	Goguen	$\begin{cases} 1, & \text{jeżeli } a = 0 \\ \min\{1, \frac{b}{a}\}, & \text{jeżeli } a > 0 \end{cases}$
7	Gödel	$\begin{cases} 1, & \text{jeżeli } a \leq b \\ b, & \text{jeżeli } a > b \end{cases}$
8	Yager	$\begin{cases} 1, & \text{jeżeli } a = 0 \\ b^a, & \text{jeżeli } a > 0 \end{cases}$
9	Zadeh	$\max\{\min\{a, b\}, 1 - a\}$
10	Willmott	$\min\left\{\frac{\max\{1 - a, b\}}{\max\{a, 1 - b, \min\{1 - a, b\}\}}\right\}$
11	Dubois–Prade	$\begin{cases} 1 - a, & \text{jeżeli } b = 0 \\ b, & \text{jeżeli } a = 1 \\ 1, & \text{w przeciwnym przypadku} \end{cases}$

c) Q -implikacje zdefiniowane w sposób następujący:

$$I(a, b) = S\{N(a), T\{a, b\}\}, \quad a, b \in [0, 1],$$

gdzie $N(a)$ jest operatorem negacji. Przykładem Q -implikacji jest implikacja Zadeha, która nie spełnia warunków a) i d) definicji 4.47.

Korzystając z tabeli 4.5, funkcje przynależności $\mu_{A \rightarrow B}(x, y)$ w przypadku modelu logicznego wyznaczamy następująco:

• **Implikacja binarna (Kleene–Dienes)**

$$\mu_{A \rightarrow B}(x, y) = \max[1 - \mu_A(x), \mu_B(y)]. \quad (4.285)$$

• **Implikacja Łukasiewicza**

$$\mu_{A \rightarrow B}(x, y) = \min[1, 1 - \mu_A(x) + \mu_B(y)]. \quad (4.286)$$

• **Implikacja Reichenbacha**

$$\mu_{A \rightarrow B}(x, y) = 1, 1 - \mu_A(x) + \mu_A(x) \cdot \mu_B(y). \quad (4.287)$$

• **Implikacja Fodora**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} 1, & \text{jeżeli } \mu_A(x) \leq \mu_B(y), \\ \max\{1 - \mu_A(x), \mu_B(y)\}, & \text{jeżeli } \mu_A(x) > \mu_B(y). \end{cases} \quad (4.288)$$

• **Implikacja Reschera**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} 1, & \text{jeżeli } \mu_A(x) \leq \mu_B(y), \\ 0, & \text{jeżeli } \mu_A(x) > \mu_B(y). \end{cases} \quad (4.289)$$

• **Implikacja Goguena**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} \min\left[1, \frac{\mu_B(y)}{\mu_A(x)}\right], & \text{jeżeli } \mu_A(x) > 0, \\ 1, & \text{jeżeli } \mu_A(x) = 0. \end{cases} \quad (4.290)$$

• **Implikacja Gödela**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} 1, & \text{jeżeli } \mu_A(x) \leq \mu_B(y), \\ \mu_B(y), & \text{jeżeli } \mu_A(x) > \mu_B(y). \end{cases} \quad (4.291)$$

• **Implikacja Yagera**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} 1, & \text{jeżeli } \mu_A(x) = 0, \\ \mu_B(y)^{\mu_A(x)}, & \text{jeżeli } \mu_A(x) > 0. \end{cases} \quad (4.292)$$

• **Implikacja Zadeha**

$$\mu_{A \rightarrow B}(x, y) = \max\{\min[\mu_A(x), \mu_B(y)], 1 - \mu_A(x)\}. \quad (4.293)$$

• **Implikacja Willmotta**

$$\mu_{A \rightarrow B}(x, y) = \min \left\{ \begin{array}{l} \max\{1 - \mu_A(x), \mu_B(y)\} \\ \max\{\mu_A(x), 1 - \mu_B(y), \min\{1 - \mu_A(x), \mu_B(y)\}\} \end{array} \right\}. \quad (4.294)$$

• **Implikacja Dubois–Prade'a**

$$\mu_{A \rightarrow B}(x, y) = \begin{cases} 1 - \mu_A(x), & \text{jeżeli } \mu_B(y) = 0, \\ \mu_B(y), & \text{jeżeli } \mu_A(x) = 1, \\ 1, & \text{w przeciwnym przypadku.} \end{cases} \quad (4.295)$$

Łatwo sprawdzić, że w przypadku modelu logicznego poszczególne reguły wnioskowania są implikacjami w sensie logicznym. Fakt ten ilustrują tabele 4.6, 4.7 i 4.8 dla implikacji binarnej, Łukasiewicza i Reichenbacha.

Tabela 4.6. Ilustracja działania implikacji binarnej

$\mu_A(x)$	$\mu_B(y)$	$1 - \mu_A(x)$	$\max[1 - \mu_A(x), \mu_B(y)]$	$\mu_{A \rightarrow B}(x, y)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Tabela 4.7. Ilustracja działania implikacji Łukasiewicza

$\mu_A(x)$	$\mu_B(y)$	$1 - \mu_A(x) + \mu_B(y)$	$\min[1, 1 - \mu_A(x) + \mu_B(y)]$	$\mu_{A \rightarrow B}(x, y)$
0	0	1	1	1
0	1	2	1	1
1	0	0	0	0
1	1	1	1	1

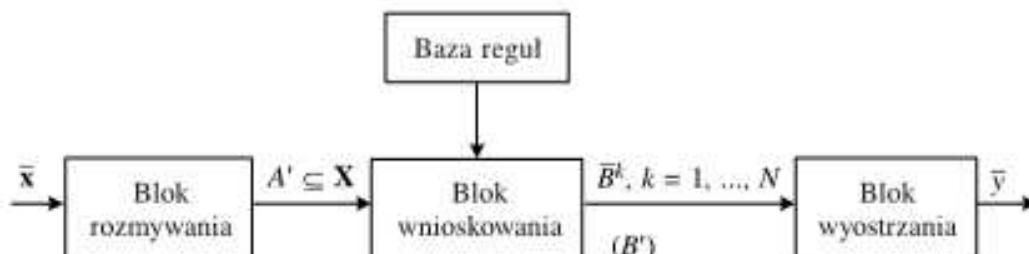
Tabela 4.8. Ilustracja działania implikacji Reichenbacha

$\mu_A(x)$	$\mu_B(y)$	$1 - \mu_A(x) + \mu_A(x) \cdot \mu_B(y)$	$1, 1 - \mu_A(x) + \mu_A(x) \cdot \mu_B(y)$	$\mu_{A \rightarrow B}(x, y)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

4.9. Rozmyte systemy wnioskujące

W wielu zagadnieniach dotyczących sterowania procesami technologicznymi niezbędne jest wyznaczenie modelu rozważanego procesu. Znajomość modelu pozwala dobrąć właściwy regulator (sterownik). Jednakże często znalezienie odpowiedniego modelu jest problemem trudnym, niekiedy wymagającym przyjęcia różnego typu założeń upraszczających. Zastosowanie teorii zbiorów rozmytych do sterowania procesami technologicznymi nie wymaga znajomości modeli tych procesów. Należy jedynie sformułować reguły postępowania w formie zdań typu **JEŻELI ... TO**. W analogiczny sposób można rozwiązać zadania klasyfikacji. Podejście wykorzystujące reguły typu **JEŻELI ... TO** pozwala rozwiązać problem klasyfikacji bez znajomości gęstości prawdopodobieństw poszczególnych klas. Sterowniki rozmyte oraz klasyfikatory są szczególnymi przypadkami rozmytych systemów wnioskujących. Na rysunku 4.33 przedstawiono typowy schemat takiego systemu. Składa się on z następujących elementów:

- 1) bazy reguł,
- 2) bloku rozmywania,
- 3) bloku wnioskowania,
- 4) bloku wyostrzania.

**Rys. 4.33.** Rozmyty system wnioskujący

W dalszych rozważaniach tego rozdziału założymy, że blok wnioskowania wykorzystuje model typu Mamdaniego, w którym poprzedniki i następcy reguł łączymy za pomocą operacji t -normy. W rozdziale 9 szczegółowo przedstawimy struktury neuronowo-rozmyte zbudowane z wykorzystaniem zarówno modelu typu Mamdaniego, jak i logicznego. Omówimy teraz poszczególne elementy rozmytego systemu wnioskującego.

4.9.1. Baza reguł

Bazę reguł, nazywaną czasami *modelem lingwistycznym*, stanowi zbiór rozmytych reguł $R^{(k)}$, $k = 1, \dots, N$, postaci

$$\begin{aligned} R^{(k)} : & \text{ JEŻELI } x_1 \text{ jest } A_1^k \text{ I } x_2 \text{ jest } A_2^k \text{ I } \dots \text{ I } x_n \text{ jest } A_n^k \\ & \text{TO } y_1 \text{ jest } B_1^k \text{ I } y_2 \text{ jest } B_2^k \text{ I } \dots \text{ I } y_m \text{ jest } B_m^k, \end{aligned} \quad (4.296)$$

gdzie N to liczba rozmytych reguł, A_i^k — zbiory rozmyte takie, że

$$A_i^k \subseteq \mathbf{X}_i \subset \mathbf{R}, \quad i = 1, \dots, n, \quad (4.297)$$

B_j^k — zbiory rozmyte takie, że

$$B_j^k \subseteq \mathbf{Y}_j \subset \mathbf{R}, \quad j = 1, \dots, m, \quad (4.298)$$

x_1, x_2, \dots, x_n — zmienne wejściowe modelu lingwistycznego, przy czym

$$[x_1, x_2, \dots, x_n]^T = \mathbf{x} \in \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n, \quad (4.299)$$

y_1, y_2, \dots, y_m — zmienne wyjściowe modelu lingwistycznego, przy czym

$$[y_1, y_2, \dots, y_m]^T = \mathbf{y} \in \mathbf{Y}_1 \times \mathbf{Y}_2 \times \dots \times \mathbf{Y}_m. \quad (4.300)$$

Symbolami \mathbf{X}_i , $i = 1, \dots, n$ oraz \mathbf{Y}_j , $j = 1, \dots, m$, oznaczamy, odpowiednio, przestrzenie zmiennych wejściowych i wyjściowych.

W toku dalszych rozważań założymy, że poszczególne reguły $R^{(k)}$, $k = 1, \dots, N$, są powiązane między sobą za pomocą operatora logicznego „lub”. Ponadto założymy, że wyjścia y_1, y_2, \dots, y_m są wzajemnie niezależne. Zatem, bez utraty ogólności, będziemy rozpatrywać rozmyte reguły ze skalarnym wyjściem postaci

$$R^{(k)} : \text{JEŻELI } x_1 \text{ jest } A_1^k \text{ I } x_2 \text{ jest } A_2^k \text{ I } \dots \text{ I } x_n \text{ jest } A_n^k \text{ TO } y \text{ jest } B^k, \quad (4.301)$$

gdzie $B^k \subseteq \mathbf{Y} \subset \mathbf{R}$ oraz $k = 1, \dots, N$. Zauważmy, że każda reguła postaci (4.301) składa się z części **JEŻELI**, nazywanej poprzednikiem (ang. *antecedent*), oraz części **TO** zwanej następcą (ang. *consequent*). Poprzednik reguły zawiera zbiór warunków, natomiast następnik zawiera wniosek. Zmienne $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ oraz y mogą przyjmować zarówno wartości określone słownie, np. „małe”, „średnie” i „duże”, jak i wartości liczbowe. Oznaczmy

$$\mathbf{X} = \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n \quad (4.302)$$

oraz

$$A^k = A_1^k \times A_2^k \times \dots \times A_n^k. \quad (4.303)$$

Stosując powyższe oznaczenia, regułę (4.301) możemy przedstawić jako rozmytą implikację

$$R^{(k)} : A^k \rightarrow B^k, \quad k = 1, \dots, N. \quad (4.304)$$

Zauważmy, że regułę $R^{(k)}$ można też interpretować jako relację rozmytą określoną na zbiorze $\mathbf{X} \times \mathbf{Y}$, tzn. $R^{(k)} \subseteq \mathbf{X} \times \mathbf{Y}$ jest zbiorem rozmytym o funkcji przynależności

$$\mu_{R^{(k)}}(\mathbf{x}, y) = \mu_{A^k \rightarrow B^k}(\mathbf{x}, y). \quad (4.305)$$

Przy projektowaniu sterowników rozmytych należy rozstrzygnąć, czy liczba reguł jest wystarczająca, czy są one niesprzeczne oraz czy zachodzą interakcje między poszczególnymi regułami. Zagadnienia te są przedmiotem szczegółowej dyskusji w pracach [33], [40] i [165].

4.9.2. Blok rozmywania

System sterowania z logiką rozmytą operuje na zbiorach rozmytych. Dlatego konkretna wartość $\bar{\mathbf{x}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n]^T \in \mathbf{X}$ sygnału wejściowego sterownika rozmytego podlega operacji rozmywania (ang. *fuzzification*), w wyniku której zostaje odwzorowana w zbiór rozmyty $A' \subseteq \mathbf{X} = \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n$. W zagadnieniach sterowania najczęściej stosuje się operację rozmywania typu singleton

$$\mu_{A'}(\mathbf{x}) = \delta(\mathbf{x} - \bar{\mathbf{x}}) = \begin{cases} 1, & \text{jeżeli } \mathbf{x} = \bar{\mathbf{x}}, \\ 0, & \text{jeżeli } \mathbf{x} \neq \bar{\mathbf{x}}. \end{cases} \quad (4.306)$$

Zbiór rozmyty A' jest wejściem bloku wnioskowania. Jeżeli sygnał wejściowy jest mierzony wraz z zakłócением, to zbiór rozmyty A' można określić poprzez funkcję przynależności

$$\mu_{A'}(\mathbf{x}) = \exp\left[-\frac{(\mathbf{x} - \bar{\mathbf{x}})^T(\mathbf{x} - \bar{\mathbf{x}})}{\delta}\right], \quad (4.307)$$

w której $\delta > 0$.

4.9.3. Blok wnioskowania

Zakładamy, że na wejściu bloku wnioskowania mamy zbiór rozmyty $A' \subseteq \mathbf{X} = \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n$. Znajdziemy odpowiedni zbiór rozmyty na wyjściu tego bloku. Rozważymy 2 przypadki, którym będą odpowiadać różne metody wyostrzania.

Przypadek 1

Na wyjściu bloku wnioskowania otrzymujemy N zbiorów rozmytych $\bar{B}^k \subseteq \mathbf{Y}$ zgodnie z uogólnioną rozmytą regułą wnioskowania *modus ponens*. Zbiór rozmyty \bar{B}^k jest określony przez złożenie zbioru rozmytego A' i relacji $R^{(k)}$, tzn.

$$\bar{B}^{(k)} = A' \circ (A^k \rightarrow B^k), \quad k = 1, \dots, N. \quad (4.308)$$

Korzystając z definicji 4.39, wyznaczamy funkcję przynależności zbioru rozmytego \bar{B}^k

$$\mu_{\bar{B}^k}(y) = \sup_{\mathbf{x} \in \mathbf{X}} [\mu_{A'}(\mathbf{x}) * \mu_{A^k \rightarrow B^k}(\mathbf{x}, y)]. \quad (4.309)$$

Konkretna postać funkcji $\mu_{\bar{B}^k}(y)$ zależy od przyjętej t -normy (podrozdz. 4.6), reguły wnioskowania (punkty 4.8.3 i 4.8.4) oraz od sposobu zdefiniowania iloczynu kartezjańskiego zbiorów rozmytych (definicja 4.36). Warto zauważyć, że w przypadku operacji rozmywania typu singleton (4.306), wzór (4.309) przybiera postać

$$\mu_{\bar{B}^k}(y) = \mu_{A^k \rightarrow B^k}(\bar{x}, y). \quad (4.310)$$

Przykład 4.40

Jeżeli $n = 2$, t -norma jest typu min, rozmyte wnioskowanie definiuje reguła typu min oraz iloczyn kartezjański zbiorów rozmytych określa wzór (4.68), to wzór (4.309) przybiera postać

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \sup_{\mathbf{x} \in \mathbf{X}} [\min(\mu_{A'}(\mathbf{x}), \mu_{A^k \rightarrow B^k}(\mathbf{x}, y))] \\ &= \sup_{\mathbf{x} \in \mathbf{X}} \{ \min [\mu_{A'}(\mathbf{x}), \min(\mu_{A'}(\mathbf{x}), \mu_{B^k}(y))] \} \\ &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} \{ \min [\mu_{A'_1}(x_1), \mu_{A'_2}(x_2), \mu_{A^k_1}(x_1), \mu_{A^k_2}(x_2), \mu_{B^k}(y)] \}. \end{aligned} \quad (4.311)$$

Ostatnia równość wynika z faktu, że

$$\mu_{A^k}(\mathbf{x}) = \mu_{A'_1 \times A'_2}(x_1, x_2) = \min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)] \quad (4.312)$$

oraz

$$\mu_{A'}(\mathbf{x}) = \mu_{A'_1 \times A'_2}(x_1, x_2) = \min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)]. \quad (4.313)$$

Przykład 4.41

Jeżeli $n = 2$, t -norma jest typu iloczyn, rozmyte wnioskowanie definiuje reguła typu iloczyn oraz iloczyn kartezjański zbiorów rozmytych określa wzór (4.69), to wzór (4.309) przybiera postać

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \sup_{\mathbf{x} \in \mathbf{X}} \{ \mu_{A'}(\mathbf{x}) \cdot \mu_{A^k \rightarrow B^k}(\mathbf{x}, y) \} \\ &= \sup_{\mathbf{x} \in \mathbf{X}} \{ \mu_{A'}(\mathbf{x}) \cdot \mu_{A^k}(\mathbf{x}) \cdot \mu_{B^k}(y) \} \\ &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} \{ \mu_{A'_1}(x_1) \cdot \mu_{A'_2}(x_2) \cdot \mu_{A^k_1}(x_1) \cdot \mu_{A^k_2}(x_2) \cdot \mu_{B^k}(y) \}. \end{aligned} \quad (4.314)$$

Przypadek 2

Na wyjściu bloku wnioskowania otrzymujemy jeden zbiór rozmyty $B' \subseteq \mathbf{Y}$, określony wzorem

$$B' = \bigcup_{k=1}^N A' \circ R^{(k)} = \bigcup_{k=1}^N A' \circ (A^k \rightarrow B^k). \quad (4.315)$$

Stosując definicję 4.35, otrzymujemy funkcję przynależności zbioru rozmytego B'

$$\mu_{B'}(y) = \bigcup_{k=1}^N \mu_{\bar{B}^k}(y), \quad (4.316)$$

przy czym funkcja przynależności $\mu_{\bar{B}^k}(y)$ jest dana wzorem (4.309).

Przykład 4.42

Rozważmy rozmyty system wnioskujący z bazą reguł:

$$R^{(1)} : \text{JEŻELI } x_1 \text{ jest } A_1^1 \text{ I } x_2 \text{ jest } A_2^1 \text{ TO } y \text{ jest } B^1, \quad (4.317)$$

$$R^{(2)} : \text{JEŻELI } x_1 \text{ jest } A_1^2 \text{ I } x_2 \text{ jest } A_2^2 \text{ TO } y \text{ jest } B^2. \quad (4.318)$$

Na wejście sterownika podano sygnał $\bar{x} = [\bar{x}_1, \bar{x}_2]^T$. W wyniku operacji rozmywania typu singleton na wejściu bloku wnioskowania otrzymujemy zbiory rozmyte A'_1 oraz A'_2 , przy czym

$$\mu_{A'_1}(x_1) = \delta(x_1 - \bar{x}_1), \mu_{A'_2}(x_2) = \delta(x_2 - \bar{x}_2). \quad (4.319)$$

Wyznaczamy sygnał wyjściowy \bar{y} sterownika rozmytego. Jako t -normę przyjmujemy operację minimum. Na podstawie wzoru (4.309) mamy

$$\mu_{\bar{B}^1}(y) = \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\min(\mu_{A'_1 \times A'_2}(x_1, x_2), \mu_{R^{(1)}}(x_1, x_2, y))]. \quad (4.320)$$

Ponadto założymy, że

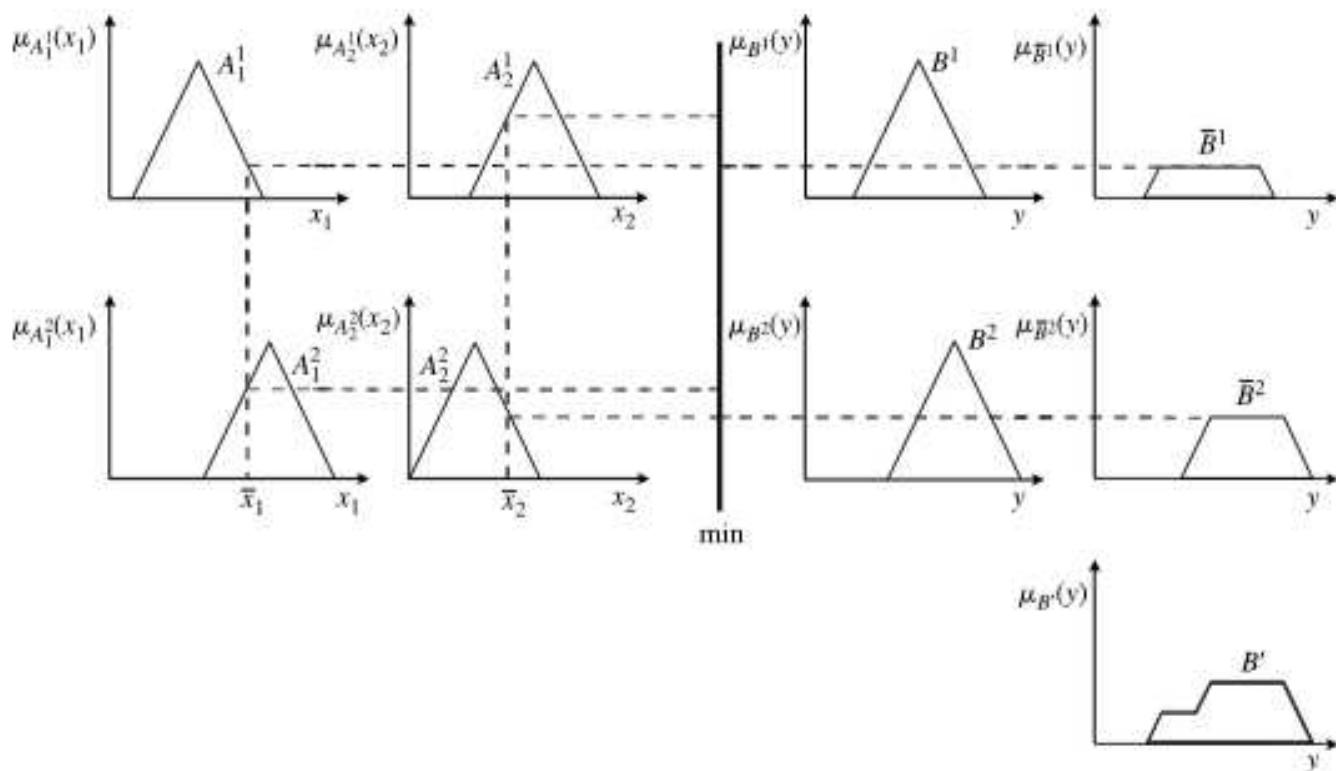
$$\mu_{A'_1 \times A'_2}(x_1, x_2) = \min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)] = \min[\delta(x_1 - \bar{x}_1), \delta(x_2 - \bar{x}_2)]. \quad (4.321)$$

Zatem

$$\begin{aligned} \mu_{\bar{B}^1}(y) &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\min(\delta(x_1 - \bar{x}_1), \delta(x_2 - \bar{x}_2), \mu_{R^{(1)}}(x_1, x_2, y))] \\ &= \mu_{R^{(1)}}(\bar{x}_1, \bar{x}_2, y) \end{aligned} \quad (4.322)$$

oraz

$$\mu_{R^{(1)}}(\bar{x}_1, \bar{x}_2, y) = \mu_{A_1^k \times A_2^k \rightarrow B^1}(\bar{x}, \bar{x}_2, y). \quad (4.323)$$



Rys. 4.34. Ilustracja do przykładu 4.42

W przypadku zastosowania reguły typu minimum (Mamdaniego) otrzymujemy

$$\mu_{A_1^k \times A_2^k \rightarrow B^k}(\bar{x}_1, \bar{x}_2, y) = \min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2), \mu_{B^k}(y)]. \quad (4.324)$$

Ponadto

$$\mu_{A_1^k \times A_2^k}(\bar{x}_1, \bar{x}_2) = \min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2)]. \quad (4.325)$$

W rezultacie

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \min\{\min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2)], \mu_{B^k}(y)\} \\ &= \min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2), \mu_{B^k}(y)] \end{aligned} \quad (4.326)$$

oraz

$$\mu_{B'}(y) = \max_{k=1,2} \{\min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2), \mu_{B^k}(y)]\}. \quad (4.327)$$

Rysunek 4.34 przedstawia graficzną interpretację rozmytego wnioskowania.

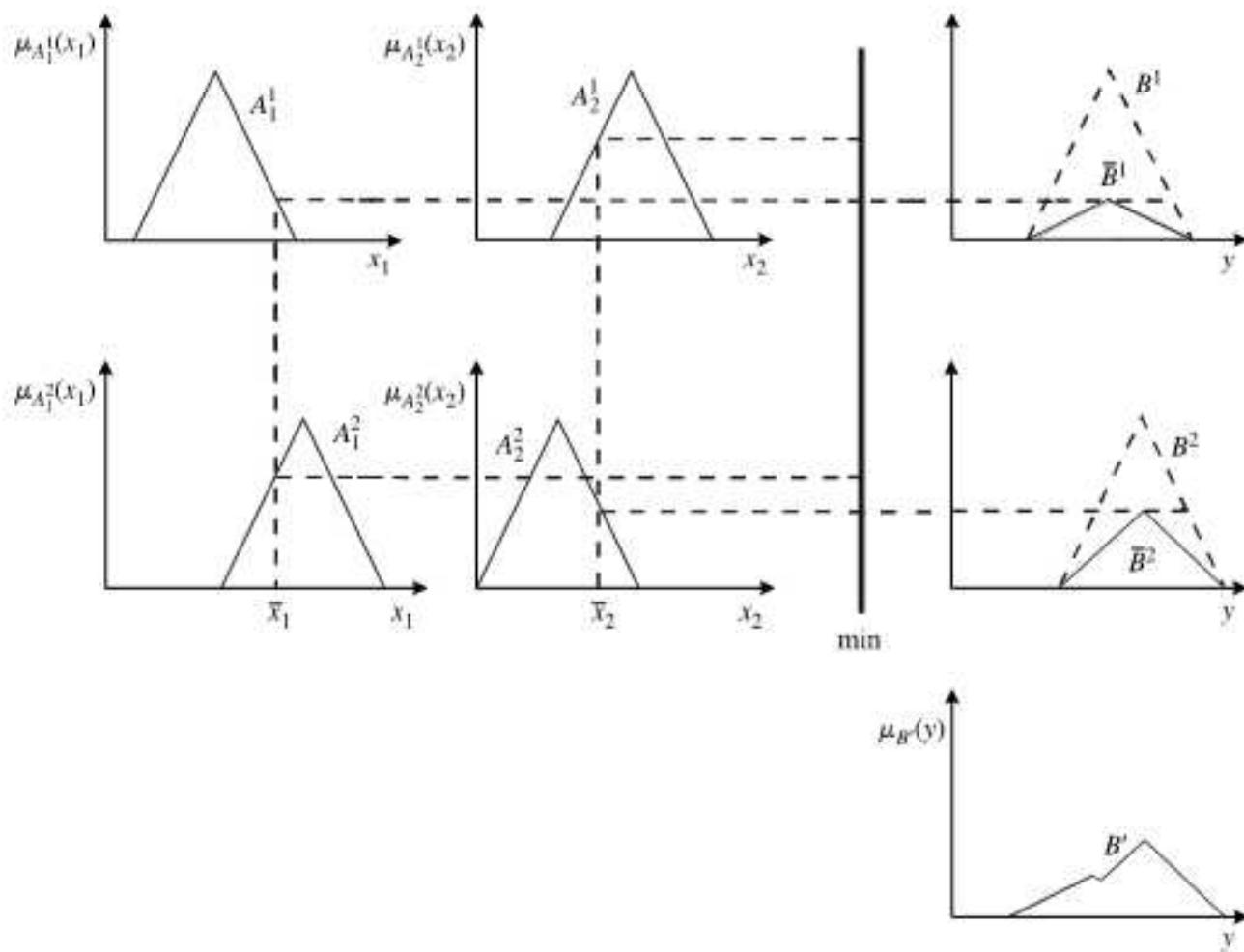
Przykład 4.43

W tym przykładzie powtóżmy rozumowanie przeprowadzone w przykładzie 4.42, ale zamiast reguły (4.324) zastosujemy regułę typu iloczyn (Larsena), tzn.

$$\mu_{A_1^k \times A_2^k \rightarrow B^k}(\bar{x}_1, \bar{x}_2, y) = \mu_{A_1^k}(\bar{x}_1) \cdot \mu_{A_2^k}(\bar{x}_2) \cdot \mu_{B^k}(y). \quad (4.328)$$

W wyniku połączenia reguł 1 i 2 otrzymamy zbiór rozmyty B' o funkcji przynależności

$$\mu_{B'}(y) = \max_{k=1,2} \{\mu_{B^k}(y) \min[\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2)]\}. \quad (4.329)$$



Rys. 4.35. Ilustracja do przykładu 4.43

W tym przypadku

$$\mu_{\bar{B}^1}(y) = \mu_{B^1}(y) \cdot \min[\mu_{A_1^1}(\bar{x}_1), \mu_{A_2^1}(\bar{x}_2)]. \quad (4.330)$$

Graficzną interpretację rozmytego wnioskowania przedstawia rysunek 4.35.

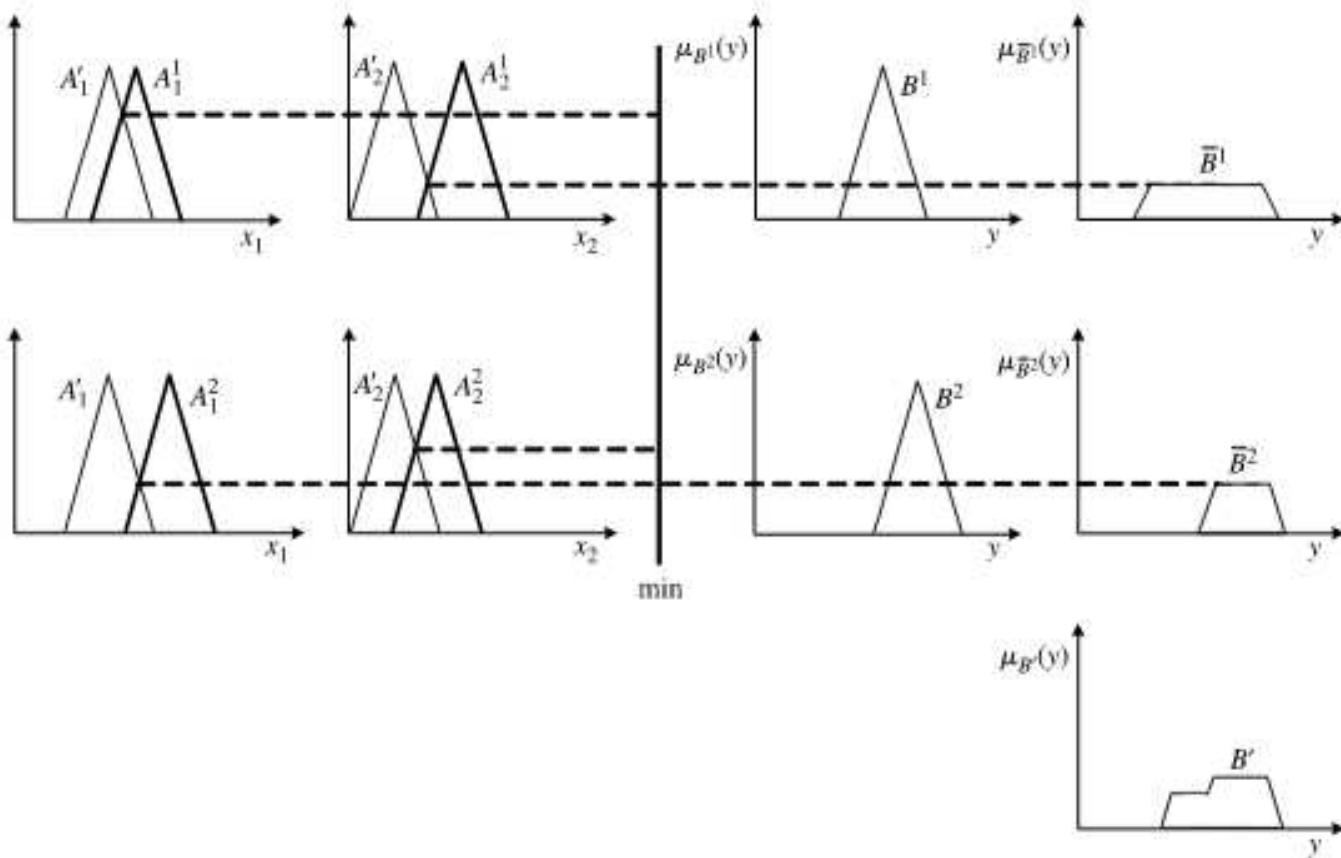
Przykład 4.44

Rozważmy rozmyty system wnioskujący opisany w przykładzie 4.42 przy założeniu, że sygnały wejściowe (numeryczne) \bar{x}_1 oraz \bar{x}_2 podlegają operacji rozmywania, w wyniku której na wejściu bloku wnioskowania otrzymujemy zbiory rozmyte A'_1 oraz A'_2 o funkcjach przynależności $\mu_{A'_1}(x_1)$ oraz $\mu_{A'_2}(x_2)$. Innymi słowy, odstępujemy od założenia (4.319) zawierającego klasę zbiorów A'_1 oraz A'_2 do rozmytych singletonów. Pozostałe założenia przyjęte w przykładzie 4.42 pozostają w mocy.

$$\begin{aligned} \mu_{\bar{B}^1}(y) &= \sup_{\mathbf{x} \in \mathbf{X}} [\mu_{A^1}(\mathbf{x}) * \mu_{R^{(1)}}(\mathbf{x}, y)] \\ &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} \min\{\min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)], \mu_{R^{(1)}}(x_1, x_2, y)\}. \end{aligned} \quad (4.331)$$

Operację min oznaczamy symbolem \wedge . Zatem

$$\begin{aligned} \mu_{\bar{B}^1}(y) &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge \mu_{R^{(1)}}(x_1, x_2, y)] \\ &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} \{\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge [(\mu_{A^1}(x_1) \wedge \mu_{A^2}(x_2)) \wedge \mu_{B^1}(y)]\} \\ &= \{\sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge (\mu_{A^1}(x_1) \wedge \mu_{A^2}(x_2))] \} \wedge \mu_{B^1}(y) \\ &= \{\sup_{x_1 \in \mathbf{X}_1} [\mu_{A'_1}(x_1) \wedge \mu_{A^1}(x_1)] \wedge \sup_{x_2 \in \mathbf{X}_2} [\mu_{A'_2}(x_2) \wedge \mu_{A^2}(x_2)]\} \wedge \mu_{B^1}(y). \end{aligned} \quad (4.332)$$



Rys. 4.36. Ilustracja do przykładu 4.44

W rezultacie

$$\mu_{B^t}(y) = \max_{k=1,2} \left\{ \left(\sup_{x_1 \in \mathbf{X}_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_1^t}(x_1)] \wedge \sup_{x_2 \in \mathbf{X}_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_2^t}(x_2)] \right) \wedge \mu_{B^t}(y) \right\}. \quad (4.333)$$

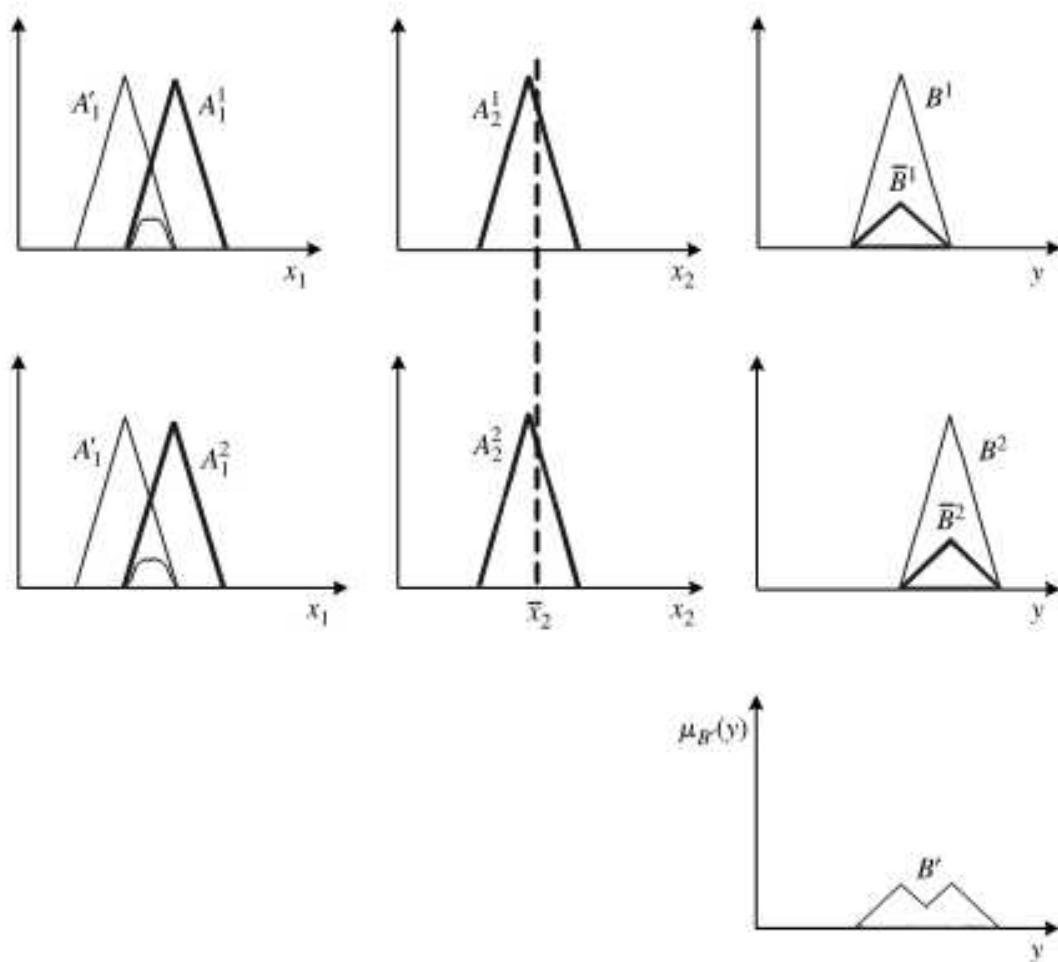
Rysunek 4.36 przedstawia graficzną interpretację rozmytego wnioskowania.

Przykład 4.45

W przykładzie 4.44 przyjęliśmy, że t -norma, iloczyn kartezjański oraz reguła wnioskowania są zdefiniowane za pomocą operacji minimum. Obecnie zastąpimy operację minimum iloczynem. Zgodnie ze wzorem (4.309) otrzymujemy

$$\begin{aligned} \mu_{\bar{B}^t}(y) &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\mu_{A'_1}(x_1) \cdot \mu_{A'_2}(x_2) \cdot \mu_{A_1^t \times A_2^t}(x_1, x_2) \cdot \mu_{B^t}(y)] \\ &= \sup_{x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2} [\mu_{A'_1}(x_1) \mu_{A'_2}(x_2) \mu_{A_1^t \times A_2^t}(x_1, x_2) \mu_{B^t}(y)] \\ &= \sup_{x_1 \in \mathbf{X}_1} [\mu_{A'_1}(x_1) \mu_{A_1^t}(x_1)] \sup_{x_2 \in \mathbf{X}_2} [\mu_{A'_2}(x_2) \mu_{A_2^t}(x_2)] \mu_{B^t}(y). \end{aligned} \quad (4.334)$$

Funkcję przynależności zbioru rozmytego B' wyznaczamy na podstawie zależności (4.334) i (4.316). Rysunek 4.37 przedstawia graficzną interpretację rozmytego wnioskowania.



Rys. 4.37. Ilustracja do przykładu 4.45

Przykład 4.46

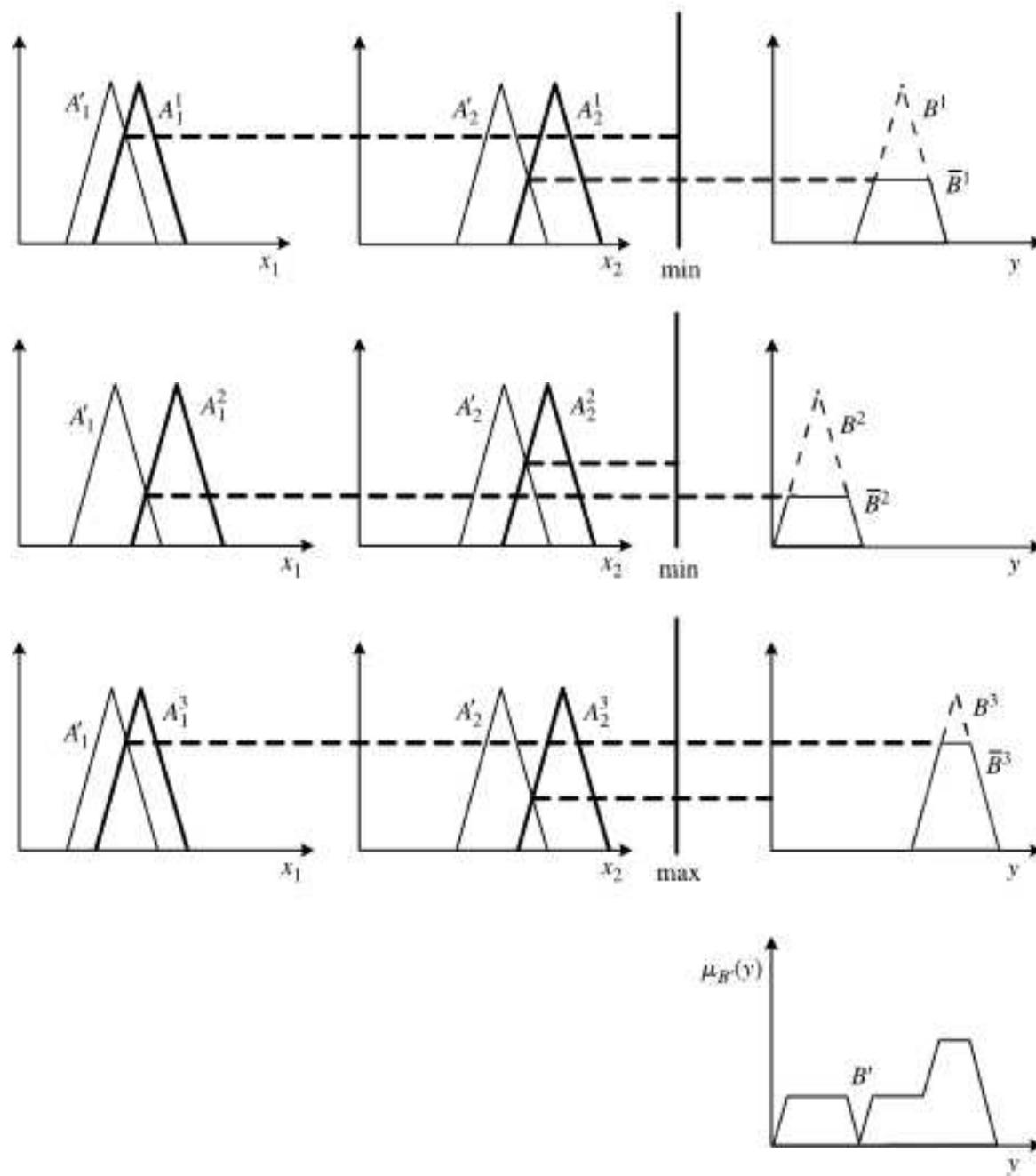
Dotychczas rozpatrywaliśmy reguły postaci (4.296). W tym przykładzie dwie pierwsze reguły $R^{(1)}$ oraz $R^{(2)}$ są szczególnymi przypadkami zapisu (4.296), natomiast reguła $R^{(3)}$ zawiera spójnik LUB:

$R^{(1)}$: **JEŻELI** x_1 jest A_1^1 **I** x_2 jest A_2^1 **TO** y jest B^1 ,

$R^{(2)}$: **JEŻELI** x_1 jest A_1^2 **I** x_2 jest A_2^2 **TO** y jest B^2 ,

$R^{(3)}$: **JEŻELI** x_1 jest A_1^3 **LUB** x_2 jest A_2^3 **TO** y jest B^3 .

Na rysunku 4.38 pokazano graficzną interpretację rozmytego wnioskowania przy założeniu, że t -norma, iloczyn kartezjański oraz reguła rozmytej implikacji są typu min. Powyższy problem można rozwiązać w inny sposób. W tym celu zauważamy, że reguły $R^{(3)}$ można zapisać w postaci dwóch reguł $\bar{R}^{(3)}$ i $\bar{R}^{(4)}$:



Rys. 4.38. Ilustracja do przykładu 4.46

$\overline{R}^{(3)}$: **JEŻELI** x_1 jest A_1^3 **TO** y jest B^3 ,

$\overline{R}^{(4)}$: **JEŻELI** x_1 jest A_2^3 **TO** y jest B^3 .

Otrzymaliśmy reguły $R^{(1)}$, $R^{(2)}$, $\overline{R}^{(3)}$ i $\overline{R}^{(4)}$, które są szczególnymi przypadkami zapisu (4.296). Czytelnik z łatwością wyprowadzi analityczną postać funkcji przynależności zbioru B' , wzorując się na wynikach przykładu 4.44.

4.9.4. Blok wyostrzania

Wielkością wyjściową bloku wnioskowania jest bądź N zbiorów rozmytych \overline{B}^k z funkcjami przynależności $\mu_{\overline{B}^k}(y)$, $k = 1, 2, \dots, N$, bądź jeden zbiór rozmyty B' z funkcją przynależności $\mu_{B'}(y)$. Pojawia się problem odwzorowania zbiorów rozmytych \overline{B}^k (lub zbioru rozmytego B') w jedną wartość $\bar{y} \in \mathbf{Y}$, która będzie wyznaczonym sterowaniem na wejściu obiektu. Odwzorowanie to nazywamy *wyostrzaniem* (ang. *defuzzification*) i jest ono realizowane w bloku wyostrzania.

Jeżeli wielkością wyjściową bloku wnioskowania jest N zbiorów rozmytych \overline{B}^k , to wartość $\bar{y} \in \mathbf{Y}$ możemy obliczyć następującymi metodami:

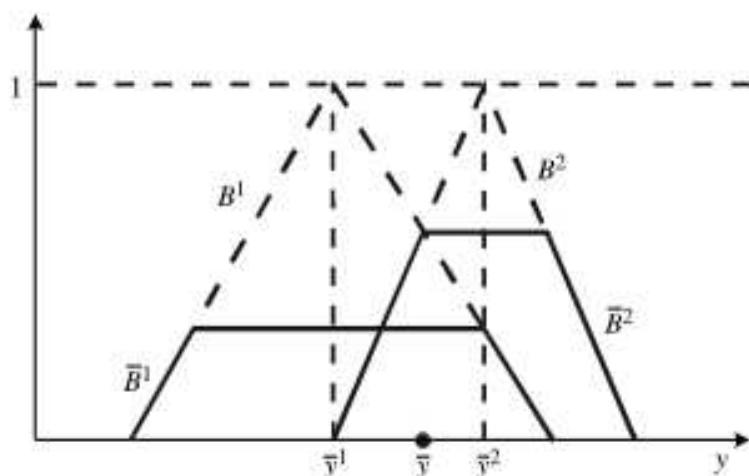
1. *Metoda center average defuzzification*. Wartość \bar{y} wyznaczamy za pomocą wzoru

$$\bar{y} = \frac{\sum_{k=1}^N \mu_{\overline{B}^k}(\bar{y}^k) \bar{y}^k}{\sum_{k=1}^N \mu_{\overline{B}^k}(\bar{y}^k)}, \quad (4.335)$$

w którym \bar{y}^k jest punktem, gdzie funkcja $\mu_{B^k}(y)$ przyjmuje wartość maximum, tzn.

$$\mu_{B^k}(\bar{y}^k) = \max_y \mu_{B^k}(y). \quad (4.336)$$

Punkt \bar{y}^k jest nazywany *środkiem* (ang. *center*) zbioru rozmytego B^k . Na rysunku 4.39 przedstawiono ideę tej metody dla $N = 2$. Zauważmy, że wartość \bar{y} nie zależy od kształtu oraz nośnika funkcji przynależności $\mu_{B^k}(y)$.



Rys. 4.39. Ilustracja metody wyostrzania *center average defuzzification*

2. *Metoda center of sums defuzzification*. Wartość \bar{y} obliczamy następująco:

$$\bar{y} = \frac{\int_Y y \sum_{k=1}^N \mu_{B^k}(y) dy}{\int_Y \sum_{k=1}^N \mu_{B^k}(y) dy}. \quad (4.337)$$

Jeżeli wielkością wyjściową bloku wnioskowania jest jeden zbiór rozmyty B' , to wartość \bar{y} możemy wyznaczyć, stosując następujące metody:

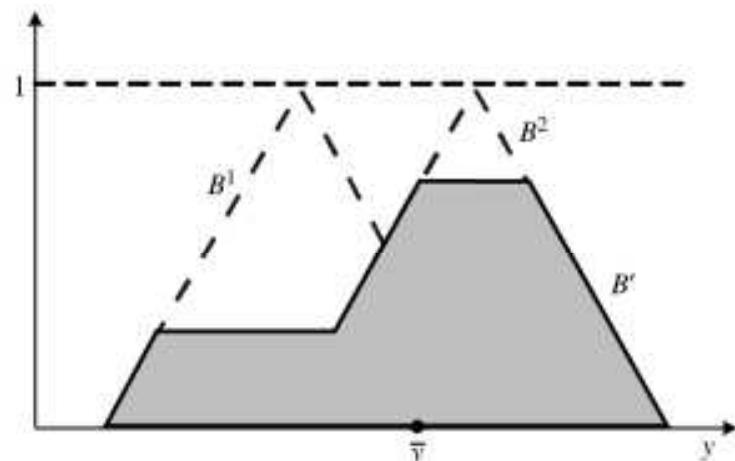
3. *Metoda środka ciężkości* (ang. *center of gravity method* lub *center of area method*). Wartość \bar{y} obliczamy jako *środek ciężkości* funkcji przynależności $\mu_{B'}(y)$, tzn.

$$\bar{y} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy} = \frac{\int_Y y S_{k=1}^N \mu_{B^k}(y) dy}{\int_Y S_{k=1}^N \mu_{B^k}(y) dy}, \quad (4.338)$$

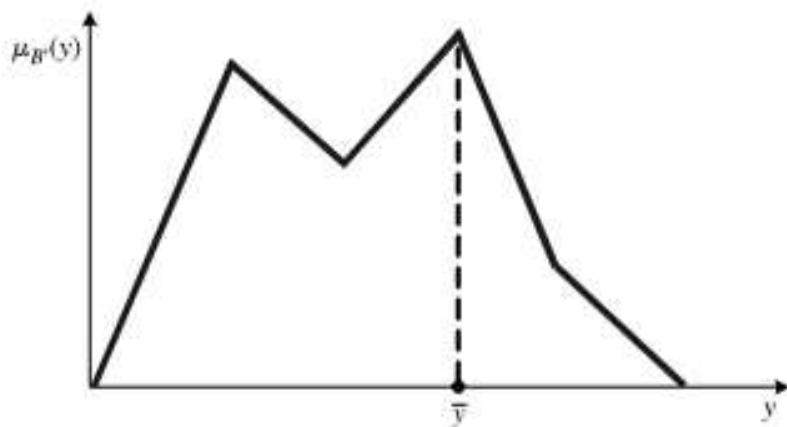
przy założeniu, że obie całki w powyższym wzorze istnieją. W przypadku dyskretnym powyższy wzór przybiera postać

$$\bar{y} = \frac{\sum_{k=1}^N \mu_{B'}(\bar{y}^k) \bar{y}^k}{\sum_{k=1}^N \mu_{B'}(\bar{y}^k)}. \quad (4.339)$$

Na rysunku 4.40 pokazano sposób wyznaczania wartości \bar{y} metodą środka ciężkości.



Rys. 4.40. Ilustracja metody środka ciężkości



Rys. 4.41. Ilustracja metody maksimum funkcji przynależności

4. *Metoda maksimum funkcji przynależności*. Wartość \bar{y} obliczamy zgodnie z zależnością

$$\mu_{B'}(\bar{y}) = \sup_{y \in Y} \mu_{B'}(y) \quad (4.340)$$

przy założeniu, że $\mu_B(y)$ jest funkcją unimodalną. Metoda ta nie uwzględnia kształtu funkcji przynależności, co ilustruje rysunek 4.41.

4.10. Zastosowania zbiorów rozmytych

4.10.1. Rozmyta metoda Delphi

Rozmyta metoda Delphi to uogólnienie klasycznej metody Delphi dotyczącej prognozowania długookresowego. Klasyczna metoda Delphi została rozwinięta w latach sześćdziesiątych ubiegłego wieku przez korporację RAN w Santa Monica w Kalifornii. Nazwa metody wywodzi się ze starożytnej Grecji od wyroczni z Delphi, słynącej z przepowiadania przyszłości. Istotę metody Delphi można opisać następująco:

- Ekspert o wysokich kwalifikacjach w danej dyscyplinie wyrażają niezależne od siebie opinie dotyczące pewnego wydarzenia w dziedzinie nauki, techniki lub biznesu. Opinie te mogą wiązać się z prognozami dotyczącymi rynku, ekonomii, postępu technologicznego itp.
- Opinie ekspertów utożsamiamy z danymi, które mają charakter subiektywny i są analizowane metodami statystycznymi. Wyznaczana jest ich wartość średnia, a wyniki są analizowane przez zarząd firmy.
- W przypadku rezultatów, które nie zadowalają zarządu firmy, ponownie prosi się ekspertów o wyrażenie opinii. Jednocześnie dostarcza się ekspertom rezultaty poprzedniej rundy zapytań.
- Proces powtarza się aż do uzyskania rozwiązania sensownego z punktu widzenia zarządu firmy. W praktyce zazwyczaj wystarczają dwie lub trzy powtórki.

W prognozowaniu długookresowym mamy do czynienia z nieścisłymi i niekompletnymi danymi. Decyzje podejmowane przez ekspertów są subiektywne i zależą w głównej mierze od ich indywidualnych kompetencji. Dlatego adekwatna wydaje się prezentacja danych za pomocą liczb rozmytych. Szczególnie odpowiednie są liczby trójkątne, dające się łatwo konstruować przez specyficzne trzy wartości: najmniejszą, największą i najbardziej prawdopodobną (w potocznym rozumieniu tego słowa). Zamiast na średniej rzeczywistej analizę opiera się na średniej rozmytej. Rozmyta metoda Delphi składa się z następujących kroków:

Krok 1. Ekspert E_i , $i = 1, \dots, n$, wyrażają swoją opinię na temat pewnego wydarzenia, np. najniższej $a_1^{(i)}$, najbardziej prawdopodobnej $a_M^{(i)}$, i najwyższej $a_2^{(i)}$ ceny euro. Informacje dostarczone przez ekspertów $E_{(i)}$ są przedstawiane przez zarząd firmy w formie rozmytych liczb trójkątnych

$$A_i = (a_1^{(i)}, a_M^{(i)}, a_2^{(i)}), \quad i = 1, \dots, n. \quad (4.341)$$

Krok 2. Obliczana jest średnia

$$A_{\text{śred}} = (m_1, m_M, m_2) = \left(\frac{1}{n} \sum_{i=1}^n a_1^{(i)}, \frac{1}{n} \sum_{i=1}^n a_M^{(i)}, \frac{1}{n} \sum_{i=1}^n a_2^{(i)} \right). \quad (4.342)$$

Krok 3. Każdy ekspert E_i ponownie wyraża swoją opinię, uwzględniając otrzymane średnie z poprzedniej rundy zapytań, i tworzone są nowe liczby rozmyte.

$$B_i = (b_1^{(i)}, b_M^{(i)}, b_2^{(i)}), \quad i = 1, \dots, n. \quad (4.343)$$

Proces powtarza się od kroku 2. Oblicza się średnią $B_{\text{śred}}$ według wzoru (4.342), z tą różnicą, że teraz $a_1^{(i)}, a_M^{(i)}, a_2^{(i)}$ jest zastępowane, odpowiednio, przez $b_1^{(i)}, b_M^{(i)}, b_2^{(i)}$. W razie potrzeby generuje się kolejne liczby trójkątne $C_i = (c_1^{(i)}, c_M^{(i)}, c_2^{(i)})$ i liczy się ich średnią $C_{\text{śred}}$. Proces powtarza się aż do osiągnięcia dwóch następujących po sobie zbliżonych średnich ($A_{\text{śred}}, B_{\text{śred}}, C_{\text{śred}}, \dots$).

Krok 4. W późniejszym czasie, jeśli pojawiły się nowe istotne informacje dotyczące danego problemu, powyższa procedura może zostać powtórzona.

Rozmyta metoda Delphi jest typową wieloekspertową procedurą prognozowania, służącą do łączenia różnych poglądów i opinii. Przedstawimy teraz dwa przykłady zastosowania rozmytej metody Delphi z wykorzystaniem trójkątnych liczb rozmytych.

Przykład 4.47. Szacunkowe wartości kursu euro w lipcu przyszłego roku

Grupę 16 ekspertów $E_i, i = 1, \dots, 16$, poproszono o wyrażenie opinii dotyczącej wartości kursu euro w lipcu przyszłego roku, stosując rozmytą metodę Delphi. Zakłada się, że opinie ekspertów mają tę samą wagę. Prognozy ekspertów, reprezentowane przez liczby trójkątne $A_i, i = 1, \dots, 16$ (4.341), zostały zamieszczone w tabeli 4.9.

Tabela 4.9. Prognozy ekspertów (pierwsze zapytanie)

Ekspert	Liczba rozmyta	Najniższa wartość	Najbardziej prawdopodobna wartość	Najwyższa wartość
E_1	A_1	$a_1^{(1)} = 3,5882$	$a_M^{(1)} = 4,2062$	$a_2^{(1)} = 4,5060$
E_2	A_2	$a_1^{(2)} = 3,9854$	$a_M^{(2)} = 4,2070$	$a_2^{(2)} = 4,6020$
E_3	A_3	$a_1^{(3)} = 3,4868$	$a_M^{(3)} = 4,2071$	$a_2^{(3)} = 4,8524$
E_4	A_4	$a_1^{(4)} = 3,9201$	$a_M^{(4)} = 4,2065$	$a_2^{(4)} = 4,7925$
E_5	A_5	$a_1^{(5)} = 4,0012$	$a_M^{(5)} = 4,2080$	$a_2^{(5)} = 4,5900$
E_6	A_6	$a_1^{(6)} = 3,8724$	$a_M^{(6)} = 4,2063$	$a_2^{(6)} = 4,9825$
E_7	A_7	$a_1^{(7)} = 3,7760$	$a_M^{(7)} = 4,2062$	$a_2^{(7)} = 4,8250$
E_8	A_8	$a_1^{(8)} = 3,8925$	$a_M^{(8)} = 4,2065$	$a_2^{(8)} = 4,6872$
E_9	A_9	$a_1^{(9)} = 3,6823$	$a_M^{(9)} = 4,2085$	$a_2^{(9)} = 4,6257$
E_{10}	A_{10}	$a_1^{(10)} = 4,0010$	$a_M^{(10)} = 4,2067$	$a_2^{(10)} = 4,6889$
E_{11}	A_{11}	$a_1^{(11)} = 3,8926$	$a_M^{(11)} = 4,2051$	$a_2^{(11)} = 4,9820$
E_{12}	A_{12}	$a_1^{(12)} = 3,5868$	$a_M^{(12)} = 4,2061$	$a_2^{(12)} = 4,9560$
E_{13}	A_{13}	$a_1^{(13)} = 3,8101$	$a_M^{(13)} = 4,2055$	$a_2^{(13)} = 4,9920$
E_{14}	A_{14}	$a_1^{(14)} = 3,7865$	$a_M^{(14)} = 4,2082$	$a_2^{(14)} = 5,0101$
E_{15}	A_{15}	$a_1^{(15)} = 3,7826$	$a_M^{(15)} = 4,2069$	$a_2^{(15)} = 4,9840$
E_{16}	A_{16}	$a_1^{(16)} = 3,7824$	$a_M^{(16)} = 4,2067$	$a_2^{(16)} = 4,7805$

Aby otrzymać średnią $A_{\text{śred}}$, najpierw sumujemy liczby w ostatnich trzech kolumnach tabeli 4.9

$$\sum_{i=1}^{16} a_1^{(i)} = 60,8469; \quad \sum_{i=1}^{16} a_M^{(i)} = 67,3075; \quad \sum_{i=1}^{16} a_2^{(i)} = 76,8568,$$

a następnie korzystamy ze wzoru (4.342)

$$A_{\text{śred}} = \left(\frac{60,8469}{16}, \frac{67,3075}{16}, \frac{76,8568}{16} \right) = (3,80293; 4,20671; 4,80355).$$

Wynik przybliżony ma postać

$$A_{\text{śred}}^p = (3,8029; 4,2067; 4,8036).$$

Latwo zauważyć, że np. opinie ekspertów E_7 i E_{16} są zbliżone do średniej $A_{\text{śred}}^p$, podczas gdy opinie ekspertów E_1 i E_{12} znacznie się od niej różnią. Założymy, że zarząd firmy postanawia powtórzyć zapytania skierowane do ekspertów, którzy otrzymują rezultaty poprzedniej rundy zapytań. Eksperci proponują nowe prognozy kursu euro, zamieniane przez zarząd firmy na liczby trójkątne B_i . Ponowne prognozy ekspertów zostały przedstawione w tabeli 4.10.

Tabela 4.10. Prognozy ekspertów (drugie zapytanie)

Ekspert	Liczba rozmyta	Najniższa wartość	Najbardziej prawdopodobna wartość	Najwyższa wartość
E_1	B_1	$b_1^{(1)} = 3,6892$	$b_M^{(1)} = 4,2060$	$b_2^{(1)} = 4,7892$
E_2	B_2	$b_1^{(2)} = 3,8026$	$b_M^{(2)} = 4,2072$	$b_2^{(2)} = 4,8020$
E_3	B_3	$b_1^{(3)} = 3,7956$	$b_M^{(3)} = 4,2060$	$b_2^{(3)} = 4,8024$
E_4	B_4	$b_1^{(4)} = 3,8026$	$b_M^{(4)} = 4,2064$	$b_2^{(4)} = 4,7824$
E_5	B_5	$b_1^{(5)} = 3,9217$	$b_M^{(5)} = 4,2050$	$b_2^{(5)} = 4,7986$
E_6	B_6	$b_1^{(6)} = 3,8056$	$b_M^{(6)} = 4,2077$	$b_2^{(6)} = 4,8008$
E_7	B_7	$b_1^{(7)} = 3,7856$	$b_M^{(7)} = 4,2066$	$b_2^{(7)} = 4,8125$
E_8	B_8	$b_1^{(8)} = 3,7985$	$b_M^{(8)} = 4,2067$	$b_2^{(8)} = 4,7892$
E_9	B_9	$b_1^{(9)} = 3,8006$	$b_M^{(9)} = 4,2079$	$b_2^{(9)} = 4,9254$
E_{10}	B_{10}	$b_1^{(10)} = 3,9121$	$b_M^{(10)} = 4,2067$	$b_2^{(10)} = 4,7986$
E_{11}	B_{11}	$b_1^{(11)} = 3,8564$	$b_M^{(11)} = 4,2066$	$b_2^{(11)} = 4,7891$
E_{12}	B_{12}	$b_1^{(12)} = 3,7859$	$b_M^{(12)} = 4,2070$	$b_2^{(12)} = 4,7682$
E_{13}	B_{13}	$b_1^{(13)} = 3,8026$	$b_M^{(13)} = 4,2065$	$b_2^{(13)} = 4,7851$
E_{14}	B_{14}	$b_1^{(14)} = 3,7998$	$b_M^{(14)} = 4,2070$	$b_2^{(14)} = 4,8102$
E_{15}	B_{15}	$b_1^{(15)} = 3,7548$	$b_M^{(15)} = 4,2067$	$b_2^{(15)} = 4,7986$
E_{16}	B_{16}	$b_1^{(16)} = 3,7266$	$b_M^{(16)} = 4,2066$	$b_2^{(16)} = 4,7256$

Stosując ponownie wzór (4.342), wyznaczamy

$$B_{\text{śred}} = (3,80258; 4,206663; 4,798619).$$

Wynik przybliżony jest następujący:

$$B_{\text{śred}}^P = (3,8026; 4,2067; 4,7986).$$

Obecnie zarząd firmy jest usatysfakcjonowany, gdyż średnie $A_{\text{śred}}^P$ i $B_{\text{śred}}^P$ są bardzo zbliżone, w związku z tym algorytm ulega zatrzymaniu, a liczba trójkątna $B_{\text{śred}}^P$ zostaje przyjęta jako wniosek łączący opinie ekspertów. Interpretacja tego wyniku jest następująca: szacunkowe wartości kursu euro w lipcu przyszłego roku mieszczą się w przedziale $[3,8026; 4,7986]$, przy czym prognozowaną ceną euro jest 4,2067 zł. Prognozę tę uzyskaliśmy poprzez wyostrzenie rozmytej liczby trójkątnej $B_{\text{śred}}^P = (3,8026; 4,2067; 4,7986)$.

4.10.2. Ważona rozmyta metoda Delphi

W wielu dziedzinach naszego życia (np. ekonomii, finansach, zarządzaniu) wiedza, doświadczenie oraz umiejętności pewnej grupy ekspertów są często stawiane wyżej niż wiedza i doświadczenie innych ekspertów. Wyraża się to za pomocą wag w_i przydzielanych ekspertom. Opiszymy teraz ważoną rozmytą metodę Delphi. Założymy, że kompetencję eksperta E_i , $i = 1, \dots, 16$, odzwierciedla waga w_i , $i = 1, \dots, 16$, $w_1 + \dots + w_n = 1$. Kolejne kroki w rozmytej metodzie Delphi będą podlegały niewielkim zmianom, a mianowicie: w kroku 2 zamiast trójkątnej średniej $A_{\text{śred}}$ pojawia się średnia ważona $A_{\text{śred}}^w$. To samo dotyczy kroku 3, gdzie zamiast średnich arytmetycznych mamy średnie ważone.

Przykład 4.48

Wróćmy do przykładu 4.47, gdzie 16 ekspertów przedstawiło swoje opinie wyrażone za pomocą liczb trójkątnych A_i zamieszczone w tabeli 4.9. Założymy teraz, że kompetencje

Tabela 4.11. Ważone prognozy ekspertów

Ekspert	w_i	$w_i \times a_1^{(i)}$	$w_i \times a_M^{(i)}$	$w_i \times a_2^{(i)}$
E_1	0,04	0,1435	0,1682	0,1802
E_2	0,04	0,1594	0,1683	0,1841
E_3	0,04	0,1395	0,1683	0,1941
E_4	0,1	0,3920	0,4207	0,4793
E_5	0,04	0,1600	0,1683	0,1836
E_6	0,04	0,1549	0,1683	0,1993
E_7	0,04	0,1510	0,1682	0,1930
E_8	0,1	0,3893	0,4207	0,4687
E_9	0,04	0,1473	0,1683	0,1850
E_{10}	0,13	0,5201	0,5469	0,6096
E_{11}	0,04	0,1557	0,1682	0,1993
E_{12}	0,04	0,1435	0,1682	0,1982
E_{13}	0,04	0,1524	0,1682	0,1997
E_{14}	0,04	0,1515	0,1683	0,2004
E_{15}	0,1	0,3783	0,4207	0,4984
E_{16}	0,13	0,4917	0,5469	0,6215
Suma:	1	3,830094	4,2067	4,79434

ekspertów E_{10}, E_{16} są oceniane najwyższej (waga 0,13), kompetencje ekspertów E_4, E_8 i E_{15} mają wagę 0,1 a pozostałych 0,04; suma wszystkich wag jest równa 1. W tabeli 4.11 przedstawiono ważone prognozy ekspertów.

Podsumowując wartości w ostatnim wierszu tabeli 4.11, otrzymujemy trójkątną średnią ważoną

$$A_{\text{sred}}^w = (3,830094; 4,2067; 4,79434).$$

Wynik przybliżony jest następujący:

$$A_{\text{sred}}^{wp} = (3,8301; 4,2067; 4,7943).$$

Uzyskany rezultat jest prawie taki sam jak w przykładzie 4.47. W wyniku wyostrzania średniej ważonej A_{sred}^{wp} uzyskujemy kwotę 4,2067.

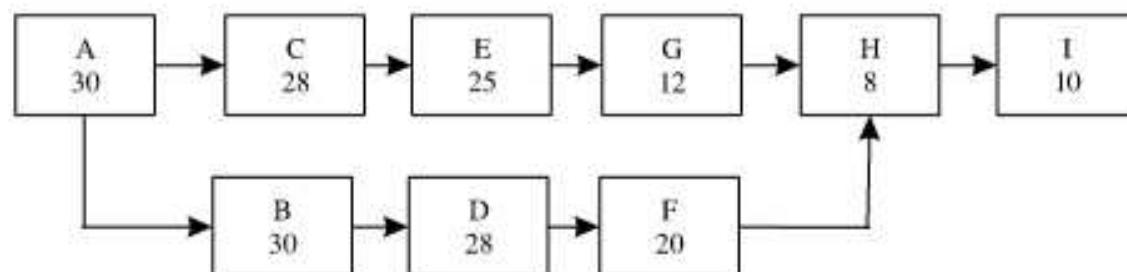
4.10.3. Rozmyta metoda PERT

Planowanie kolejności działań jest skomplikowanym przedsięwzięciem wymagającym uwzględnienia różnych czynności, które mają być wykonywane w procesie projektowania nowego produktu lub technologii. W drugiej połowie lat pięćdziesiątych ubiegłego wieku w USA zaproponowano dwie nowe metody organizowania dużych, złożonych przedsięwzięć produkcyjnych lub budowlanych, w których uczestniczy wielu kooperantów i współwykonawców. W literaturze metody te znane są jako metoda PERT (ang. *Project Evaluation and Review Technique*) oraz metoda ścieżki krytycznej (ang. *Critical Path Method* — CPM). Techniki PERT i CPM są podobne do siebie i często używane razem jako jedna metoda. Po raz pierwszy metoda PERT została zastosowana w USA w 1957 r. w związku z budową atomowych okrętów podwodnych i rakiet Polaris. Natomiast metodę CPM stosowano mniej więcej w tym samym czasie w ośrodkach badawczych firm Remington Rand i DuPont zajmujących się strategią planowania w fabrykach chemicznych. Metoda PERT polega na stworzeniu modelu przebiegu (sieci) czynności prowadzących do realizacji założonego celu, z uwzględnieniem czasu trwania każdej z nich (najkrótszego, najdłuższego i najbardziej prawdopodobnego). Ciąg czynności, z których każda kolejna jest uwarunkowana wykonaniem poprzedniej, tworzy tzw. ścieżkę krytyczną wyznaczającą najdłuższy czas realizacji zadania. Jego skrócenie może nastąpić przez wyłączenie ze ścieżki krytycznej tych czynności, które mogą być równolegle wykonywane, oraz przez przyspieszenie wykonania pozostałych. W rozwiniętej postaci tej metody uwzględnia się także koszty realizacji poszczególnych czynności (etapów) zadania. Metoda PERT pozwala na skonstruowanie harmonogramu prac optymalizującego czas i koszty realizacji zadania, pozwalającego na bezkolizyjną współpracę wszystkich jego uczestników, wyeliminowanie przestojów i tzw. wąskich gardeł. Działanie metody PERT przedstawimy na uproszczonym przykładzie projektowania procesu produkcyjnego sprzętu AGD. Projekt danego urządzenia wymaga zaprojektowania, wyprodukowania i montażu poszczególnych elementów oraz przetestowania gotowego produktu. W naszym przykładzie projekt złożony jest z dziewięciu różnych działań A, B, C, D, E, F, G, H, I. Wymagany czas zakończenia każdego działania zamieszczono w ostatniej kolumnie tabeli 4.12. Został on oszacowany przez menedżerów odpowiedzialnych za poszczególne działania.

Tabela 4.12. Ważone prognozy ekspertów

	Rodzaj działania	Działania poprzedzające	Działania równolegle	Działania następne	Wymagany czas zakończenia (dni)
A	Projektowanie mechaniki	—	—	B, C	30
B	Projektowanie instalacji elektrycznej	A	C, E	D	30
C	Produkcja części mechanicznych	A	B	E	28
D	Produkcja części elektrycznych	B	E, G	F	28
E	Montaż części mechanicznych	C	B, D	G	25
F	Montaż części elektrycznych	D, E	—	H	20
G	Montaż części elektronicznych	E	—	H	12
H	Uruchamianie	G, F	—	I	8
I	Testy	H	—	—	10

Najpierw skonstruujemy sieciowy model planowania uwzględniający dane zwarte w tabeli 4.12. Model ten został przedstawiony na rysunku 4.42. Każde działanie jest reprezentowane przez prostokąt, w którego środku umieszczamy symbol rodzaju działania wraz z liczbą dni potrzebnych na jego wykonanie. Sieciowy model planowania przedstawia sekwencyjne powiązanie między działaniami. *Ścieżkę krytyczną* definiujemy jako ciąg działań w kolejności od początkowego do ostatniego w projekcie, wymagający najdłuższego czasu ukończenia. W związku z tym, całkowity wymagany czas wykonania projektu jest tożsamy z czasem potrzebnym do zakończenia działań znajdujących się na ścieżce krytycznej.



Rys. 4.42. Sieciowy model planowania

Sieciowy model planowania pomaga wyznaczyć ścieżkę krytyczną, która na rysunku 4.42 została przedstawiona za pomocą bloczków połączonych strzałkami łączącymi działania A, B, D, F, H, I. Zatem całkowity czas ukończenia projektu wynosi:

$30 + 30 + 28 + 20 + 8 + 10 = 126$ dni. Na rysunku 4.42 zauważamy, że działania C, E oraz G nie znajdują się na ścieżce krytycznej. Dlatego ich wykonanie może przedłużyć się ponad $28 + 25 + 12 = 65$ dni, jednak opóźnienie nie może być większe niż 13 dni, aby nie spowodować wydłużenia ścieżki krytycznej.

W naszym przykładzie czas każdego rodzaju działania będzie oszacowany przez trzech ekspertów. Zadaniem ekspertów jest oszacowanie optymistycznego, najbardziej prawdopodobnego (w potocznym rozumieniu tego słowa) oraz pesymistycznego czasu ukończenia zadań A, B, ..., I. Opinie ekspertów zostały zapisane w postaci trójkątnych liczb rozmytych $T_i^A, T_i^B, \dots, T_i^I, i = 1, 2, 3$. Tabela 4.13 przedstawia opinie ekspertów dotyczące czasu wykonania zadania A.

Tabela 4.13. Szacunkowy czas zakończenia zadania A

Ekspert	T_i^A	Czas optymistyczny	Czas najbardziej prawdopodobny	Czas pesymistyczny
E_1	T_1^A	28	29	33
E_2	T_2^A	27	30	32
E_3	T_3^A	27	31	34
Suma	$\sum_{i=1}^3 T_i^A$	82	90	99

Średni czas wykonania zadania A reprezentuje rozmyta liczba trójkątna postaci

$$T_{\text{śred}}^A = \left(\frac{82}{3}, \frac{90}{3}, \frac{99}{3} \right) = (27,3; 30; 33).$$

Rzeczywisty czas zakończenia zadania otrzymamy w wyniku wyostrzania liczby trójkątnej $T_{\text{śred}}^A$. W zależności od zastosowanej metody wyostrzania (podrozdz. 4.5) otrzymujemy następujące rezultaty:

$$y^{(1)} = 30,$$

$$y^{(2)} = \frac{27,3 + 30 + 33}{3} = 30,1,$$

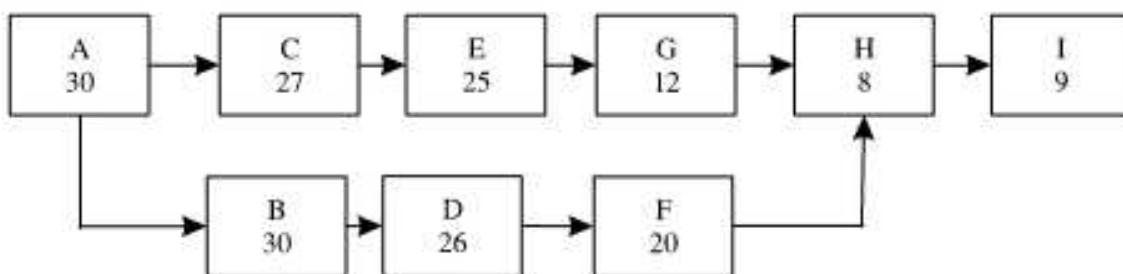
$$y^{(3)} = \frac{27,3 + 2(30) + 33}{4} = 30,075,$$

$$y^{(4)} = \frac{27,3 + 4(30) + 33}{6} = 30,05.$$

Podobnie, osiem innych trzyosobowych grup ekspertów wyraziło swoje opinie dotyczące oszacowania wymaganego czasu zakończenia poszczególnych zadań. W tabeli 4.14 przedstawione zostały średnie czasy $T_{\text{śred}}^B, \dots, T_{\text{śred}}^I$ (czas $T_{\text{śred}}^A$ jest również wzięty pod uwagę). Każda liczba trójkątna w tabeli 4.14, przedstawiająca średni czas danego zadania, zostaje wyostrzona (operacja max) w celu otrzymania rzeczywistego czasu zakończenia tego zadania. Na rysunku 4.43 przedstawiono sieciowy model planowania z uwzględnieniem opinii ekspertów.

Tabela 4.14. Średni czas zakończenia poszczególnych zadań

Zadanie	Średni czas działania	Czas optymistyczny t_1	Czas najbardziej prawdopodobny t_M	Czas pesymistyczny t_2
A	$T_{śred}^A$	27	30	33
B	$T_{śred}^B$	28	30	32
C	$T_{śred}^C$	24	27	31
D	$T_{śred}^D$	24	26	29
E	$T_{śred}^E$	22	25	27
F	$T_{śred}^F$	17	20	23
G	$T_{śred}^G$	10	12	14
H	$T_{śred}^H$	6	8	11
I	$T_{śred}^I$	7	9	12



Rys. 4.43. Zmodyfikowany sieciowy model planowania

Całkowity czas potrzebny na ukończenie projektu wynosi

$$T = T_{śred}^A + T_{śred}^B + T_{śred}^D + T_{śred}^F + T_{śred}^H + T_{śred}^I = (109, 123, 140).$$

W związku z tym czas trwania projektu waha się między 109 a 140 dniami, przy czym najbardziej prawdopodobny czas, według opinii ekspertów, to 123 dni.

4.10.4. Podejmowanie decyzji w otoczeniu rozmytym

Teoria zbiorów rozmytych pozwala na podejmowanie decyzji w tzw. *otoczeniu rozmytym*, które składa się z celów rozmytych, ograniczeń rozmytych i decyzji rozmytej. Rozważmy pewien zbiór opcji (zwanych również wyborami lub wariantami) oznaczony przez $X_{op} = \{x\}$. *Cel rozmyty* definiuje się jako zbiór rozmyty G określony w zborze opcji X_{op} . Zbiór rozmyty G opisany jest funkcją przynależności $\mu_G : X_{op} \rightarrow [0, 1]$. Funkcja $\mu_G(x) \in [0, 1]$ dla konkretnego x określa stopień przynależności opcji $x \in X_{op}$ do zbioru rozmytego G (cel rozmyty). *Ograniczenie rozmyte* definiuje się jako zbiór rozmyty C również określony w zborze opcji X_{op} . Zbiór rozmyty C opisany jest funkcją przynależności $\mu_C : X_{op} \rightarrow [0, 1]$. Funkcja $\mu_C(x) \in [0, 1]$ dla konkretnego x określa stopień przynależności opcji $x \in X_{op}$ do zbioru rozmytego C (ograniczenie rozmyte). Rozważmy zadanie wyznaczenia decyzji jednocześnie osiągającej cel rozmyty G i spełniającej ograniczenie rozmyte C . *Decyzja rozmyta* D jest zbiorem rozmytym powstały w wyniku przecięcia celu rozmytego i ograniczenia rozmytego:

$$D = G \cap C, \quad (4.344)$$

przy czym

$$\mu_D(x) = T\{\mu_G(x), \mu_C(x)\} \quad (4.345)$$

dla każdego $x \in X$. Warto zauważyć, że zapis (4.345) sugeruje następującą interpretację zadania podejmowania decyzji w otoczeniu rozmytym: „osiągnąć G i spełnić C ”. Konkretna postać wzoru (4.345) zależy od przyjętej t -normy.

Powyższe rozważania łatwo uogólnić na przypadek wielu celów i ograniczeń. Założymy, że mamy $n > 1$ celów rozmytych, G_1, \dots, G_n , i $m > 1$ ograniczeń rozmytych, C_1, \dots, C_m , a wszystkie są określone jako zbiory rozmyte w zbiorze opcji X_{op} . Decyzję rozmytą wyznacza się w sposób następujący:

$$D = G_1 \cap \dots \cap G_n \cap C_1 \cap \dots \cap C_m, \quad (4.346)$$

przy czym

$$\mu_D(x) = T\{\mu_{G_1}(x), \dots, \mu_{G_n}(x), \mu_{C_1}(x), \dots, \mu_{C_m}(x)\} \quad (4.347)$$

dla każdego $x \in X_{\text{op}}$. Decyzją maksymalizującą jest opcja $x^* \in X$, taka że

$$\mu_D(x^*) = \max_{x \in X} \mu_D(x). \quad (4.348)$$

Powyższe rozważania zilustrujemy na przykładach konkretnych zastosowań podejmowania decyzji w otoczeniu rozmytym:

- a) podział dywidendy,
- b) polityka zatrudnienia,
- c) polityka mieszkaniowa dla rodzin z niskim dochodem,
- d) ocena studentów,
- e) strategia wyboru uczelni,
- f) ustalanie ceny nowego produktu.

We wszystkich przykładach wyznaczymy decyzję rozmytą typu minimum, tzn. t -norma we wzorze (4.345) jest postaci \min .

Przykład 4.49. Podział dywidendy

Walne zgromadzenie akcjonariuszy po zatwierdzeniu bilansu spółki rozpatruje wysokość dywidendy przypadającej na jedną akcję. Wysokość dywidendy jest zmienną lingwistyczną przyjmującą dwie wartości: *atrakcyjna dywidenda* i *umiarkowana dywidenda*. Wartość lingwistyczna *atrakcyjna dywidenda* jest celem opisywanym przez zbiór rozmyty G , zdefiniowany na zbiorze opcji $X_{\text{op}} = \{x : 0 < x \leq 70\}$, gdzie opcja x jest wyrażona w złotówkach. Funkcja przynależności $\mu_G(x)$ jest rosnąca. Wartość lingwistyczna *umiarkowana dywidenda* jest ograniczeniem opisywanym przez zbiór rozmyty C , zdefiniowany na tym samym przedziale X_{op} z malejącą funkcją przynależności $\mu_C(x)$. Zakładamy, że zbiór rozmyty G *atrakcyjna dywidenda* jest postaci

$$\mu_G(x) = \begin{cases} 0 & \text{dla } 0 < x \leq 10, \\ \frac{1}{40}x - \frac{1}{4} & \text{dla } 10 \leq x \leq 50, \\ 1 & \text{dla } 50 \leq x \leq 70, \end{cases}$$

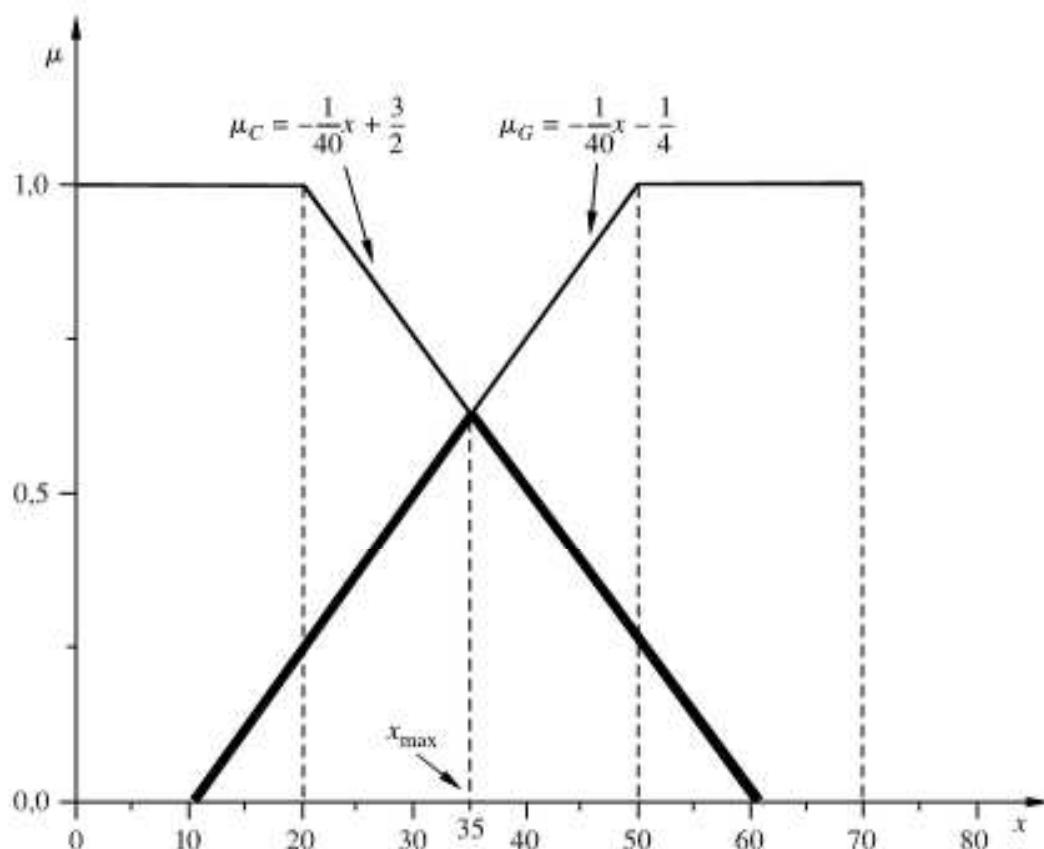
natomiast zbiór rozmyty C umiarkowana dywidenda jest dany na X_{op} w sposób następujący:

$$\mu_C(x) = \begin{cases} 1 & \text{dla } 0 < x \leq 20, \\ -\frac{1}{40}x + \frac{3}{2} & \text{dla } 20 \leq x \leq 60, \\ 0 & \text{dla } 60 \leq x \leq 70. \end{cases}$$

Decyzja rozmyta typu minimum jest postaci

$$\mu_D(x) = \min(\mu_G(x), \mu_C(x)),$$

co przedstawia rysunek 4.44. Punktem przecięcia funkcji $\mu_G(x) = \frac{1}{40}x - \frac{1}{4}$ i $\mu_C(x) = -\frac{1}{40}x + \frac{3}{2}$ jest $(35; 0,625)$. Zatem $x^* = 35$ i wysokość wypłacanej dywidendy wynosi 35 zł.



Rys. 4.44. Cel G , ograniczenie C i decyzja D

Przykład 4.50. Polityka zatrudnienia

Rozważmy firmę, która ogłosiła konkurs na stanowisko asystenta dyrektora. Podczas rozmowy kwalifikacyjnej kandydaci pytani są o rodzaj posiadanych kwalifikacji, doświadczenie, wiedzę w danej dziedzinie itp. W naszym przypadku ustalono następujące kryteria oceny kandydatów:

- G_1 — doświadczenie,
- G_2 — znajomość obsługi komputera,
- G_3 — młody wiek,
- G_4 — znajomość języka obcego.

Celem firmy jest znalezienie na stanowisko asystenta dyrektora najlepszego kandydata, który zaakceptuje oferowane przez przedsiębiorstwo wynagrodzenie. Poszczególni kan-

dydaci x_k , $k = 1, \dots, 4$ oceniani są z punktu widzenia osiągnięcia celów G_1 , G_2 , G_3 i G_4 oraz spełnienia ograniczenia C . Wynikiem tej oceny są zbiory rozmyte zdefiniowane na zbiorze $X_{\text{op}} = \{x_1, x_2, x_3, x_4\}$

$$G_1 = \frac{0,7}{x_1} + \frac{0,2}{x_2} + \frac{0,5}{x_3} + \frac{0,3}{x_4},$$

$$G_2 = \frac{0,8}{x_1} + \frac{0,8}{x_2} + \frac{0,5}{x_3} + \frac{0,2}{x_4},$$

$$G_3 = \frac{0,7}{x_1} + \frac{0,8}{x_2} + \frac{0,4}{x_3} + \frac{0,5}{x_4},$$

$$G_4 = \frac{0,5}{x_1} + \frac{0,6}{x_2} + \frac{0,7}{x_3} + \frac{0,8}{x_4}.$$

Ograniczenie C oznacza gotowość kandydatów do zaakceptowania oferowanego przez firmę wynagrodzenia i zdaniem firmy przedstawia się następująco:

$$C = \frac{0,3}{x_1} + \frac{0,4}{x_2} + \frac{0,6}{x_3} + \frac{0,9}{x_4}.$$

Decyzję D wyznaczamy na podstawie następującego wzoru:

$$D = G_1 \cap G_2 \cap G_3 \cap G_4 \cap C.$$

W wyniku prostych obliczeń otrzymujemy

$$D = \frac{0,3}{x_1} + \frac{0,2}{x_2} + \frac{0,4}{x_3} + \frac{0,2}{x_4}.$$

Kandydat nr 3 charakteryzuje się największym stopniem przynależności równym 0,4, w związku z tym jest on najlepszym kandydatem na oferowane przez firmę stanowisko. Przedstawiony w naszym przykładzie model decyzyjny dla polityki zatrudnienia może być stosowany w podobnych sytuacjach.

Przykład 4.51. Polityka mieszkaniowa dla rodzin z niskim dochodem

Rada miejska zamierza wprowadzić politykę mieszkaniową dla rodzin z niskimi dochodami, zamieszkujących stare budownictwo zlokalizowane na dużych działkach. Rozpatruje się trzy alternatywne projekty:

x_1 — remont i zarządzanie budynkami,

x_2 — program TBS (Towarzystwo Budownictwa Społecznego),

x_3 — preferencyjny kredyt na zakup nowego mieszkania.

Zbiorem opcji jest $X_{\text{op}} = \{x_1, x_2, x_3\}$. Projekty te będą wymagały częściowego lub pełnego przeniesienia rodzin. Rada miejska po zapoznaniu się z opiniami ekspertów zaproponowała trzy cele i dwa ograniczenia opisane przez zbiory rozmyte zdefiniowane na X_{op} . Przedstawiają się one następująco:

$$\text{„Poprawa standardu życia mieszkańców"} = G_1 = \frac{0,7}{x_1} + \frac{0,8}{x_2} + \frac{0,9}{x_3},$$

$$\text{„Większa liczba mieszkań na tym samym terenie"} = G_2 = \frac{0}{x_1} + \frac{0,9}{x_2} + \frac{0,9}{x_3},$$

$$\text{„Lepsze warunki mieszkaniowe"} = G_3 = \frac{0,3}{x_1} + \frac{0,7}{x_2} + \frac{0,8}{x_3},$$

$$\text{„Rozsądny koszt"} = C_1 = \frac{0,8}{x_1} + \frac{0,6}{x_2} + \frac{0,3}{x_3},$$

$$\text{„Krótki czas realizacji"} = C_2 = \frac{0,9}{x_1} + \frac{0,2}{x_2} + \frac{0,7}{x_3}.$$

Decyzję rozmytą wyznaczamy w sposób następujący:

$$D = G_1 \cap G_2 \cap G_3 \cap C_1 \cap C_2.$$

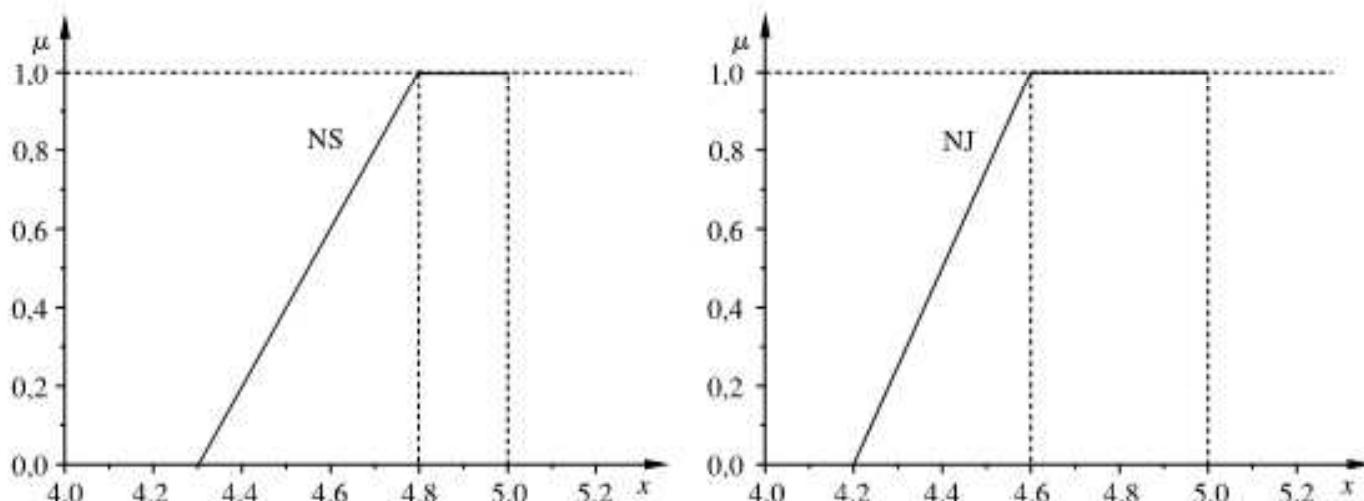
Dla t -normy typu minimum otrzymujemy

$$D = \frac{0}{x_1} + \frac{0,2}{x_2} + \frac{0,3}{x_3}.$$

Projekt x_3 z największym stopniem przynależności 0,3 okazał się najlepszym rozwiązaniem.

Przykład 4.52. Ocena studentów

Firma ufundowała wakacyjne praktyki dla studentów, którzy uzyskali najlepsze wyniki z przedmiotów ścisłych (elektronika, informatyka, matematyka) oraz z języków (angielski, niemiecki). Słowo *najlepszy* to wartość lingwistyczna, którą opisano oddziennie dla przedmiotów ścisłych (NS) i języków (NJ) i przedstawiono na rysunku 4.45, przyjmując skalę ocen [2, 5].



Rys. 4.45. Funkcje przynależności zbiorów rozmytych NS i NJ

Funkcje przynależności zbiorów rozmytych NS i NJ są następujące:

$$\mu_{NS}(x) = \begin{cases} 0 & \text{dla } 2 \leq x \leq 4,3; \\ \frac{x - 4,3}{0,5} & \text{dla } 4,3 \leq x \leq 4,8; \\ 1 & \text{dla } 4,8 \leq x \leq 5, \end{cases} \quad (4.349)$$

oraz

$$\mu_{NJ}(x) = \begin{cases} 0 & \text{dla } 2 \leq x \leq 4,2; \\ \frac{x - 4,2}{0,4} & \text{dla } 4,2 \leq x \leq 4,6; \\ 1 & \text{dla } 4,6 \leq x \leq 5. \end{cases} \quad (4.350)$$

Studentom, którzy uzyskali z przedmiotów ścisłych średnią 4,8 i wyższą, przyporządkowuje się stopień przynależności równy 1. W przypadku języków analogiczna wartość średnia wynosi 4,6. W naszym przykładzie sześciu studentów ($x_1 = \text{Kasia}$, $x_2 = \text{Małgosia}$, $x_3 = \text{Ania}$, $x_4 = \text{Tomek}$, $x_5 = \text{Jacek}$, $x_6 = \text{Michał}$) ubiega się o miejsce na wakacyjnych praktykach. Zbiorem opcji jest $X_{\text{op}} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$. W tabeli 4.15 zostały zamieszczone średnie ocen studentów z poszczególnych przedmiotów.

Tabela 4.15. Średnia ocen studentów z poszczególnych przedmiotów

Student	Elektronika	Informatyka	Matematyka	angielski	Niemiecki
Kasia (x_1)	4,8	5,0	4,7	4,3	4,7
Małgosia (x_2)	4,4	4,7	4,8	4,4	4,4
Ania (x_3)	4,9	4,9	4,6	4,7	4,3
Tomek (x_4)	4,5	4,8	4,9	5,0	4,5
Jacek (x_5)	5,0	4,6	4,7	4,4	5,0
Michał (x_6)	4,9	4,5	5,0	4,5	4,4

Podstawiając średnią ocen studentów z przedmiotów ścisłych do wzoru (4.349), otrzymujemy stopnie przynależności do zbioru rozmytego NS. Analogicznie, podstawiając średnią ocen studentów z języków obcych do wzoru (4.350) otrzymujemy stopnie przynależności do zbioru rozmytego NJ.

Tabela 4.16. Wartości stopni przynależności do zbiorów rozmytych NS i NJ

Student	Elektronika	Informatyka	Matematyka	angielski	Niemiecki
Kasia (x_1)	1	1	0,8	0,25	1
Małgosia (x_2)	0,2	0,8	1	0,5	0,5
Ania (x_3)	1	1	0,6	1	0,25
Tomek (x_4)	0,4	1	1	1	0,75
Jacek (x_5)	1	0,6	0,8	0,5	1
Michał (x_6)	1	0,4	1	0,75	0,5

Kolejnym krokiem jest stworzenie zbiorów rozmytych korespondujących z danymi zawartymi w tabeli 4.16.

$$\text{„Najlepszy z elektroniki”} = G_1 = \frac{1}{x_1} + \frac{0,2}{x_2} + \frac{1}{x_3} + \frac{0,4}{x_4} + \frac{1}{x_5} + \frac{1}{x_6},$$

$$\text{„Najlepszy z informatyki”} = G_2 = \frac{1}{x_1} + \frac{0,8}{x_2} + \frac{1}{x_3} + \frac{1}{x_4} + \frac{0,6}{x_5} + \frac{0,4}{x_6},$$

$$\text{„Najlepszy z matematyki”} = G_3 = \frac{0,8}{x_1} + \frac{1}{x_2} + \frac{0,6}{x_3} + \frac{1}{x_4} + \frac{0,8}{x_5} + \frac{1}{x_6},$$

$$\text{„Doskonały z angielskiego”} = G_4 = \frac{0,25}{x_1} + \frac{0,5}{x_2} + \frac{1}{x_3} + \frac{1}{x_4} + \frac{0,5}{x_5} + \frac{0,75}{x_6},$$

$$\text{„Doskonaly z niemieckiego”} = G_5 = \frac{1}{x_1} + \frac{0,5}{x_2} + \frac{0,25}{x_3} + \frac{0,75}{x_4} + \frac{1}{x_5} + \frac{0,5}{x_6}.$$

Podstawiając dane do wzoru (4.346), otrzymujemy

$$D = G_1 \cap G_2 \cap G_3 \cap G_4 \cap G_5.$$

Decyzja rozmyta typu minimum jest postaci

$$D = \frac{0,25}{x_1} + \frac{0,2}{x_2} + \frac{0,25}{x_3} + \frac{0,4}{x_4} + \frac{0,5}{x_5} + \frac{0,4}{x_6}.$$

Student x_5 charakteryzuje się największym stopniem przynależności, a zatem on pojedzie na letnie praktyki.

Przykład 4.53. Strategia wyboru uczelni

Zdolny uczeń złożył dokumenty w kilku uczelniach i po zdaniu egzaminów został przyjęty do 4 szkół, które tworzą zbiór opcji $X_{op} = \{x_1, x_2, x_3, x_4\}$. Teraz podejmie decyzję, do której szkoły będzie uczęszczał. Celem ucznia jest uczenie się w renomowanej uczelni (tj. znajdującej się w czołówce rankingu najlepszych szkół wyższych). Jednocześnie przyszły student chciałby, aby spełnione były pewne warunki, a mianowicie: szkoła powinna znajdować się w niezbyt dużej odległości od miejsca zamieszkania; w szkole powinien istnieć program wymiany międzynarodowej; powinna ona posiadać dobre zaplecze techniczne oraz po jej ukończeniu uczeń chce mieć duże szanse na znalezienie pracy. Warunki te zostały zapisane za pomocą zbiorów rozmytych:

$$\text{„Niezbyt duża odległość od miejsca zamieszkania”} = C_1 = \frac{0,8}{x_1} + \frac{0,9}{x_2} + \frac{0,4}{x_3} + \frac{0,5}{x_4},$$

$$\text{„Program wymiany międzynarodowej”} = C_2 = \frac{0,2}{x_1} + \frac{0,2}{x_2} + \frac{0,9}{x_3} + \frac{0,6}{x_4},$$

„Dobre zaplecze techniczne na uczelni (wyposażenie sal, laboratoriów itp.)”

$$= C_3 = \frac{0,5}{x_1} + \frac{0,3}{x_2} + \frac{0,6}{x_3} + \frac{0,7}{x_4},$$

$$\text{„Duże możliwości znalezienia pracy”} = C_4 = \frac{0,6}{x_1} + \frac{0,5}{x_2} + \frac{0,7}{x_3} + \frac{0,7}{x_4}.$$

W tabeli 4.17 przypisano stopnie przynależności poszczególnym uczelniom (gdzie x_2 , ze stopniem przynależności równym 1, to uczelnia zajmująca pierwsze miejsce w rankingu itd.).

Tabela 4.17. Szkoły wyższe wraz z przypisanymi im stopniami przynależności

Uczelnia	x_1	x_2	x_3	x_4
Stopień przynależności miejsca w rankingu	0,75	1	0,25	0,5

Wykorzystując dane zawarte w tabeli 4.17, tworzymy zbiór rozmyty G opisujący cel

$$G = \frac{0,75}{x_1} + \frac{1}{x_2} + \frac{0,25}{x_3} + \frac{0,5}{x_4}.$$

Korzystając ze wzoru (4.346), otrzymujemy następującą decyzję rozmytą:

$$D = G \cap C_1 \cap C_2 \cap C_3 \cap C_4.$$

Decyzja rozmyta typu minimum jest postaci

$$D = \frac{0,2}{x_1} + \frac{0,2}{x_2} + \frac{0,25}{x_3} + \frac{0,5}{x_4}.$$

Największy stopień przynależności wynosi 0,5, a zatem uczeń wybierze uczelnię x_4 .

Przykład 4.54. Ustalenie ceny nowego produktu

Ustalenie ceny nowego produktu wchodzącego na rynek to skomplikowany proces. Wymaga on wspólnego wysiłku ekspertów z takich dziedzin, jak finanse, zarządzanie, marketing i sprzedaż. Zadaniem ekspertów jest ustalenie ceny produktu. Typowe wymagania dotyczące ustalenia ceny nowego produktu są następujące:

W_1 — produkt powinien mieć *niską cenę*,

W_2 — produkt powinien mieć *wysoką cenę*,

W_3 — produkt powinien mieć *cenę bliską konkurencyjnej*,

W_4 — produkt powinien mieć *cenę bliską podwójnej ceny wytworzenia*.

Rozważmy model cenowy składający się z trzech wymagań W_1, W_3, W_4 . Założymy, że cena konkurencyjna wynosi 35 zł, a podwojony koszt wytworzenia wynosi 40 zł. Zbiór opcji X_{op} mieści się w przedziale $[20, 60]$, co oznacza, że cena produktu powinna być zawarta w tym przedziale. Model ten został przedstawiony na rysunku 4.46. Wartości lingwistyczne w poszczególnych wymaganiach są opisane przez zbiory rozmyte następująco: wymaganie W_1 jest wyrażone przez rozmytą liczbę trójkątną A_1 (*niska cena*), wymagania W_3 i W_4 zostały przedstawione za pomocą rozmytych liczb trójkątnych A_3 (*zblżone do ceny konkurencyjnej*) i A_4 (*bliskie podwójnego kosztu wytworzenia*). Funkcje przynależności trójkątnych liczb rozmytych A_1, A_3 i A_4 są następujące:

$$\mu_{A_1}(x) = \begin{cases} \frac{-x + 50}{30} & \text{dla } 20 \leq x \leq 50, \\ 0 & \text{w przeciwnym przypadku;} \end{cases}$$

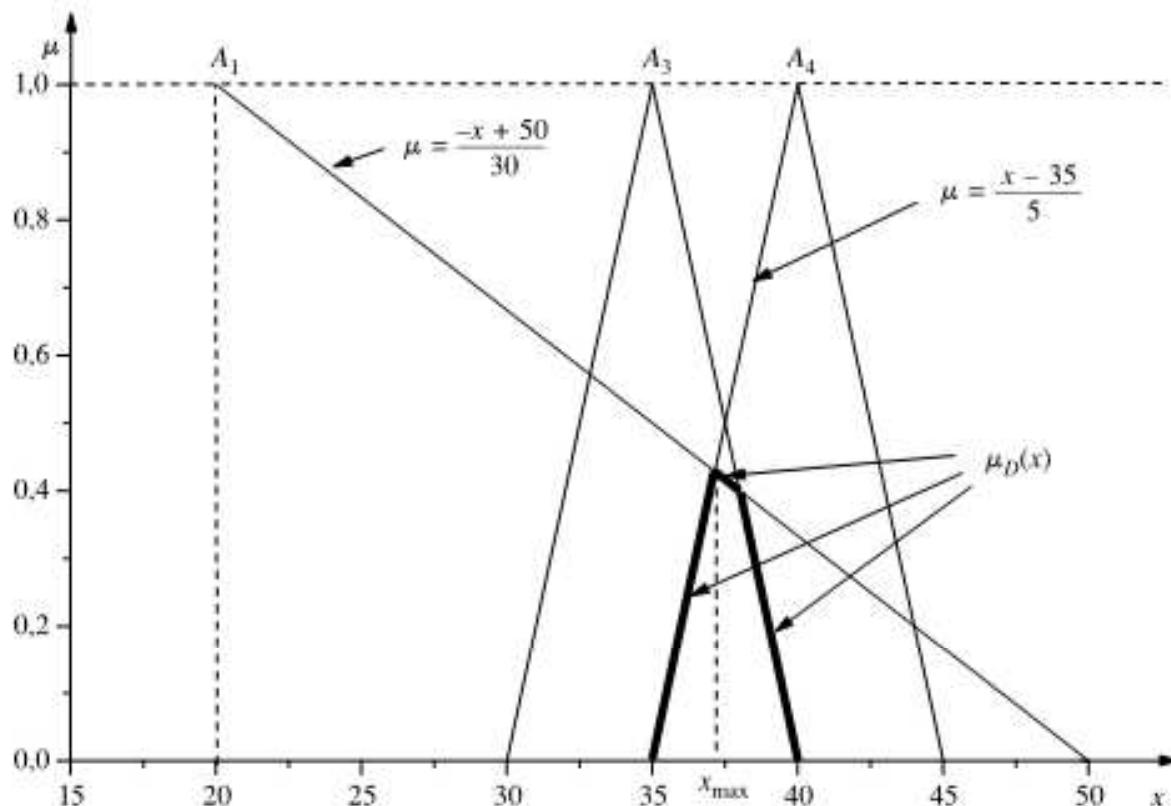
$$\mu_{A_3}(x) = \begin{cases} \frac{x - 30}{5} & \text{dla } 30 \leq x \leq 35, \\ \frac{-x + 40}{5} & \text{dla } 35 \leq x \leq 40, \\ 0 & \text{w przeciwnym przypadku;} \end{cases}$$

$$\mu_{A_4}(x) = \begin{cases} \frac{x - 35}{5} & \text{dla } 35 \leq x \leq 40, \\ \frac{-x + 45}{5} & \text{dla } 40 \leq x \leq 45, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Rozmyta decyzja D typu minimum jest postaci

$$\mu_D(x) = \min(\mu_{A_1}(x), \mu_{A_3}(x), \mu_{A_4}(x)).$$

Znajdując punkt przecięcia prostych $\mu = \frac{-x+50}{30}$ i $\mu = \frac{x-35}{5}$, otrzymujemy decyzję $x^* = 37,14$ interpretowaną jako cena produktu. Ekspertci akceptują tę cenę jako zalecaną. Można zauważyć na rysunku 4.46, że rozmyta liczba trójkątna A_3 (*bliska ceny konkurencyjnej*) ma wpływ na rozmytą decyzję D , ale nie ma wpływu na decyzję maksymalizującą x^* . Jedynie rozmyta liczba trójkątna A_4 (*bliska podwójnego kosztu wytworzenia*) i A_1 (*niska cena*) mają wpływ na wartość x^* .



Rys. 4.46. Model cenowy z wymaganiami W_1, W_3, W_4

Przykład 4.55. Ustalenie ceny nowego produktu

Kontynuujemy przykład 4.54, modyfikując wymaganie W_1 w sposób następujący:

W_1 — produkt powinien mieć *bardzo niską cenę*. Pozostałe wymagania W_2, W_3 i W_4 nie ulegają zmianom.

Zgodnie ze wzorem (4.75) funkcja przynależności *bardzo A₁* ma postać

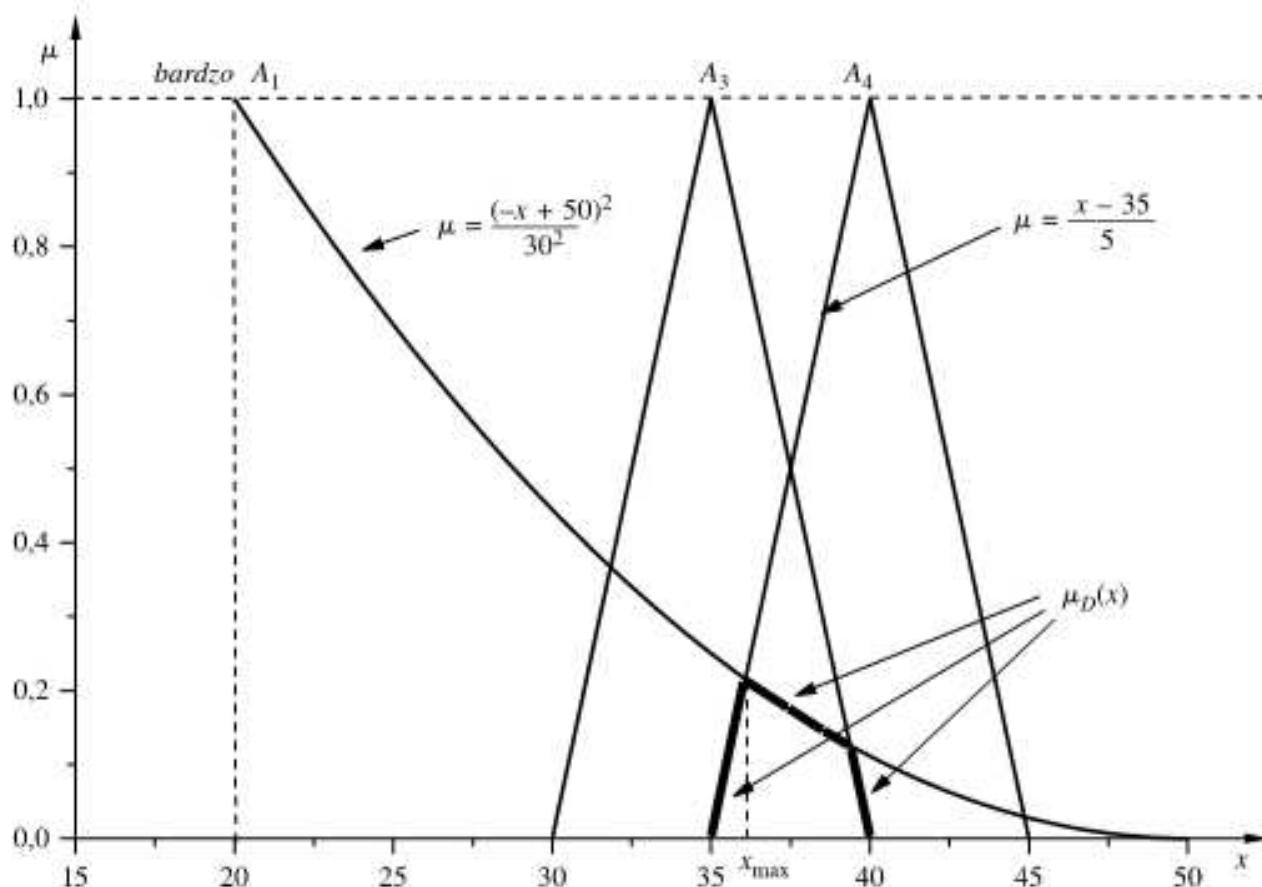
$$\mu_{\text{bardzo } A_1}(x) = (\mu_{A_1}(x))^2 = \begin{cases} \left(\frac{-x+50}{30}\right)^2 & \text{dla } 20 \leq x \leq 50, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Jest to parabola zdefiniowana w przedziale $[20, 50]$, pokazana na rysunku 4.47.

Decyzję rozmytą D wyznaczamy następująco:

$$\mu_D(x) = \min(\mu_{\text{bardzo } A_1}(x), \mu_{A_3}(x), \mu_{A_4}(x)).$$

Aby znaleźć x^* , należy wyznaczyć punkt przecięcia funkcji $\mu = \left(\frac{-x+50}{30}\right)^2$ oraz $\mu = \frac{x-35}{5}$. Otrzymujemy równanie kwadratowe $x^2 - 280x + 8800 = 0$, którego rozwiązania to 36,075



Rys. 4.47. Model cenowy z wymaganiami bardzo W_1, W_3, W_4

i 243,925. Rozwiążanie znajdujące się w przedziale $[35, 40]$, $x^* = 36,075 \approx 36$, daje sugerowaną cenę produktu. Modyfikator *bardzo* daje większy nacisk na *niską cenę*, dlatego otrzymaliśmy 36, czyli cenę niższą niż 37,14 uzyskaną w poprzednim przykładzie. Podobnie jak w przykładzie 4.54, liczba trójkątna A_3 (*cena bliska konkurencyjnej*) przyczynia się do rozmytej decyzji D , ale nie ma wpływu na wartość x^* .

Przykład 4.56. Ustalenie ceny nowego produktu

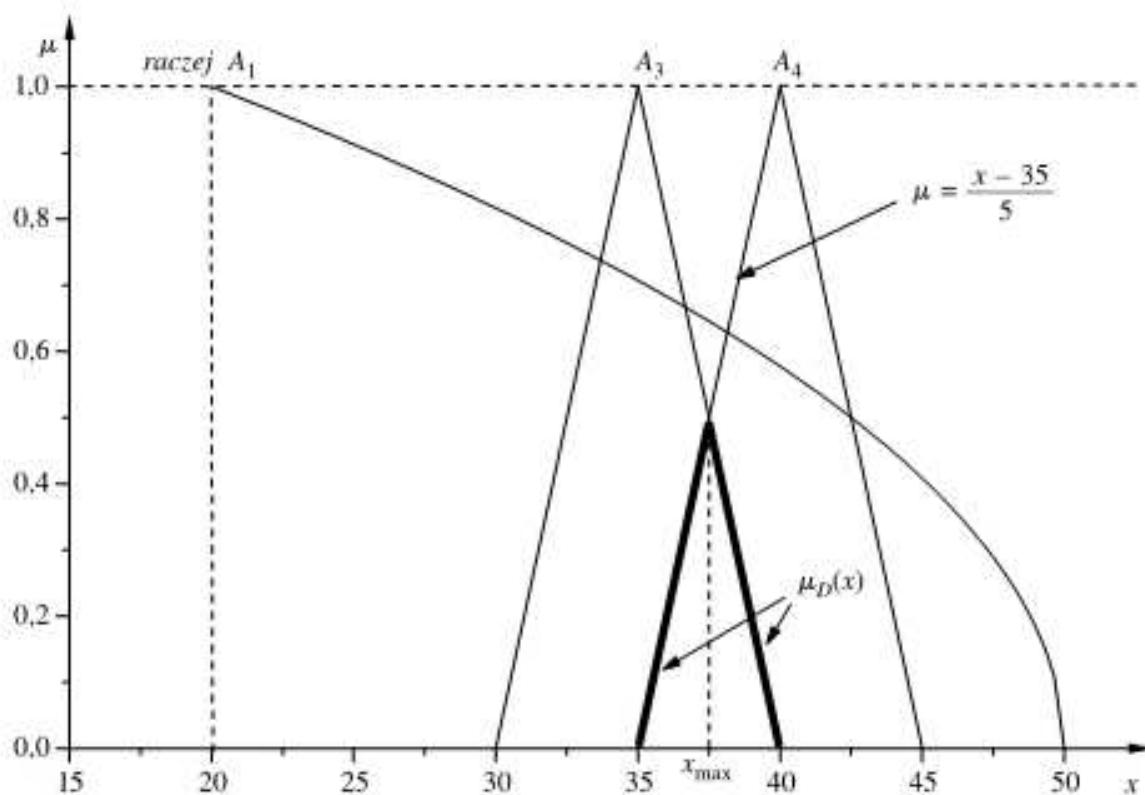
Obecnie w przykładzie 4.54 modyfikujemy wymaganie W_1 w sposób następujący: W_1 — produkt powinien mieć *raczej niską cenę*. Pozostałe wymagania W_2, W_3 i W_4 nie ulegają zmianom.

Zgodnie ze wzorem (4.76) funkcja przynależności *raczej* A_1 ma postać

$$\mu_{\text{raczej } A_1}(x) = (\mu_{A_1}(x))^{\frac{1}{2}} = \begin{cases} \left(\frac{-x + 50}{30}\right)^{\frac{1}{2}} & \text{dla } 20 \leq x \leq 50, \\ 0 & \text{w przeciwnym przypadku,} \end{cases}$$

czyli jest parabolą w przedziale $[20, 50]$ przedstawioną na rysunku 4.48. Z rysunku 4.48 wynika, że wymaganie W_1 (*raczej niska cena*) nie wpływa na podjęcie rozmytej decyzji D charakteryzującej się funkcją przynależności $\mu_D(x)$, przy czym $x^* = 37,5$.

Zauważmy, że modele kształtuowania cen w przykładach 4.54 i 4.55 prowadzą do decyzji respektujących *niską cenę* i *cenę bliską podwójnej ceny wytworzenia* bez zastanawiania się nad *ceną konkurencyjną*. Przedsiębiorstwo z taką polityką cenową może tworzyć przychylny rynek dla konkurencji, ale firma może narazić się na straty lub na eliminację z rynku.

Rys. 4.48. Model cenowy z wymaganiami *raczej* W_1, W_3, W_4

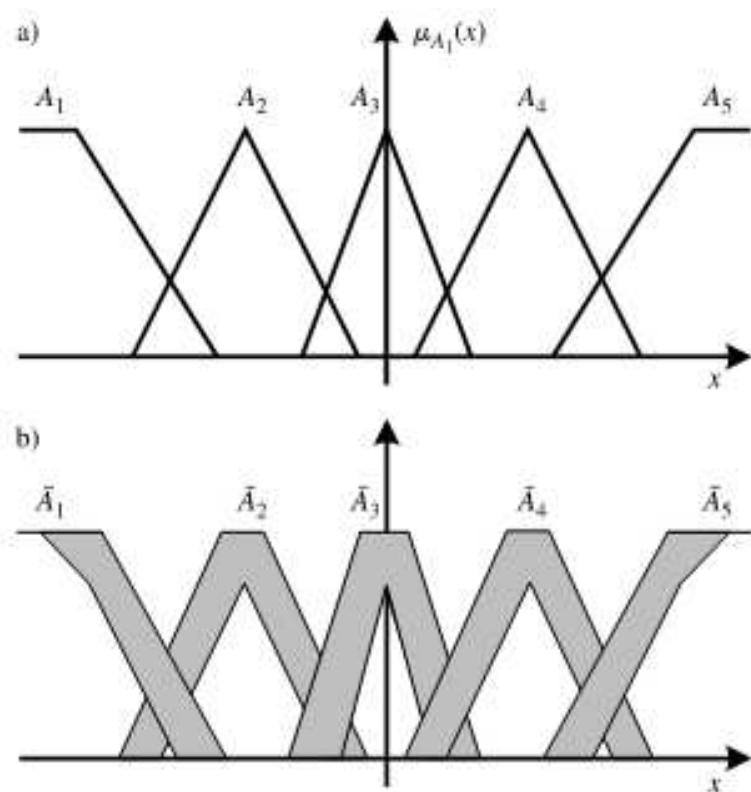
4.11. Uwagi

Teorię zbiorów rozmytych zaproponował w roku 1965 Lotfi Zadeh [265]. Podstawy tej teorii oraz liczne jej zastosowania przedstawiono w wielu monografiach [17, 33, 79, 94, 118, 130, 165–167, 269]. Na szczególne wyróżnienie zasługuje obszerna monografia Piegata [171]. Zagadnień projektowania rozmytych systemów wnioskujących do celów sterowania dotyczą monografie [3, 33, 39, 40, 253]. Godna polecenia jest znakomita monografia Kacprzyka [95] na temat podejmowania decyzji w warunkach rozmytych oraz wieloetapowego sterowania rozmytego. W monografii [13] autorzy podali rozliczne zastosowania zbiorów rozmytych w ekonomii i zarządzaniu. Analogiczne przykłady przedstawiliśmy w podrozdziale 4.10. Różne typy generatorów norm trójkątnych oraz ich właściwości omówiono w monografiach [111, 144]. W pracach [43, 44, 87, 150, 180] autorzy podejmują różne wątki dotyczące połączenia teorii zbiorów rozmytych z teorią zbiorów przybliżonych. W monografiach [116] i [119] przedstawiono zastosowania zbiorów rozmytych w zagadnieniach diagnostyki procesów przemysłowych. W pracy [252] podano sposób generacji reguł rozmytych na podstawie ciągu uczącego. W monografii [21] autorzy przedstawiają różne typy operatorów mających zastosowanie m.in. do agregacji reguł rozmytych.

Metody reprezentacji wiedzy z wykorzystaniem zbiorów rozmytych typu 2

5.1. Wprowadzenie

Zbiory rozmyte rozważane w poprzednim rozdziale nazywamy zbiorami rozmytymi typu 1. Są one scharakteryzowane poprzez funkcję przynależności, przy czym wartość tej funkcji dla danego elementu x nazywamy stopniem przynależności tego elementu do zbioru rozmytego. W przypadku zbiorów rozmytych typu 1 stopień przynależności jest liczbą rzeczywistą przyjmującą wartości w przedziale $[0, 1]$. W tym rozdziale przedstawimy inną koncepcję rozmytego opisu niepewności. Według tej koncepcji stopień przynależności nie jest już liczbą, lecz ma charakter rozmyty. Na rysunku 5.1 przedstawiono ilustrację graficzną zbiorów rozmytych A_1, \dots, A_5 typu 1 oraz odpowiadające im zbiory rozmyte $\tilde{A}_1, \dots, \tilde{A}_5$ typu 2. Zauważmy, że w przypadku zbiorów rozmytych typu 2, dla dowolnego ustalonego elementu x , nie możemy mówić o jednoznacznie określonej wartości funkcji przynależności. Innymi słowy, stopień przynależności nie jest liczbą, tak jak w przypadku zbiorów rozmytych typu 1.



Rys. 5.1. Zbiory rozmyte: a) typu 1, b) typu 2

W kolejnych punktach rozdziału przedstawimy podstawowe definicje dotyczące zbiorów rozmytych typu 2, omówimy operacje na tych zbiorach, a następnie relacje rozmyte typu 2 oraz metody redukcji typu, czyli sposoby transformacji zbiorów rozmytych typu 2 w zbiory rozmyte typu 1.

W ostatniej części rozdziału teoria zbiorów rozmytych typu 2 posłuży do konstrukcji rozmytego systemu wnioskującego. Omówione zostaną szczegółowo poszczególne bloki takiego systemu, włączając w to rozmywanie typu 2, bazę reguł typu 2, mechanizmy wnioskowania typu 2 oraz dwuetapowe wyostrzanie, na które składa się redukcja typu i właściwe wyostrzanie.

5.2. Podstawowe definicje

DEFINICJA 5.1

Zbiorem rozmytym \tilde{A} typu 2, określonym na przestrzeni rozważań X , co oznaczamy $\tilde{A} \subseteq X$, nazywamy zbiór par

$$\{x, \mu_{\tilde{A}}(x)\}, \quad (5.1)$$

gdzie x jest elementem zbioru rozmytego, a jego stopień przynależności $\mu_{\tilde{A}}(x)$ do zbioru rozmytego \tilde{A} jest zbiorem rozmytym typu 1, określonym na przedziale $J_x \subseteq [0, 1]$, tzn.

$$\mu_{\tilde{A}}(x) = \int_{u \in J_x} f_x(u)/u. \quad (5.2)$$

Funkcję $f_x : [0, 1] \rightarrow [0, 1]$ będziemy nazywać *funkcją drugorzędnej przynależności*, a jej wartość $f_x(u)$ *stopniem drugorzędnej przynależności* lub krócej *drugorzędną przynależnością*. Oczywiście u jest argumentem funkcji drugorzędnej przynależności. Przedział J_x , będący dziedziną funkcji drugorzędnej przynależności f_x , jest nazywany *podstawową przynależnością* elementu x . Zbiór rozmyty \tilde{A} możemy zapisać w notacji zbiorów rozmytych następująco:

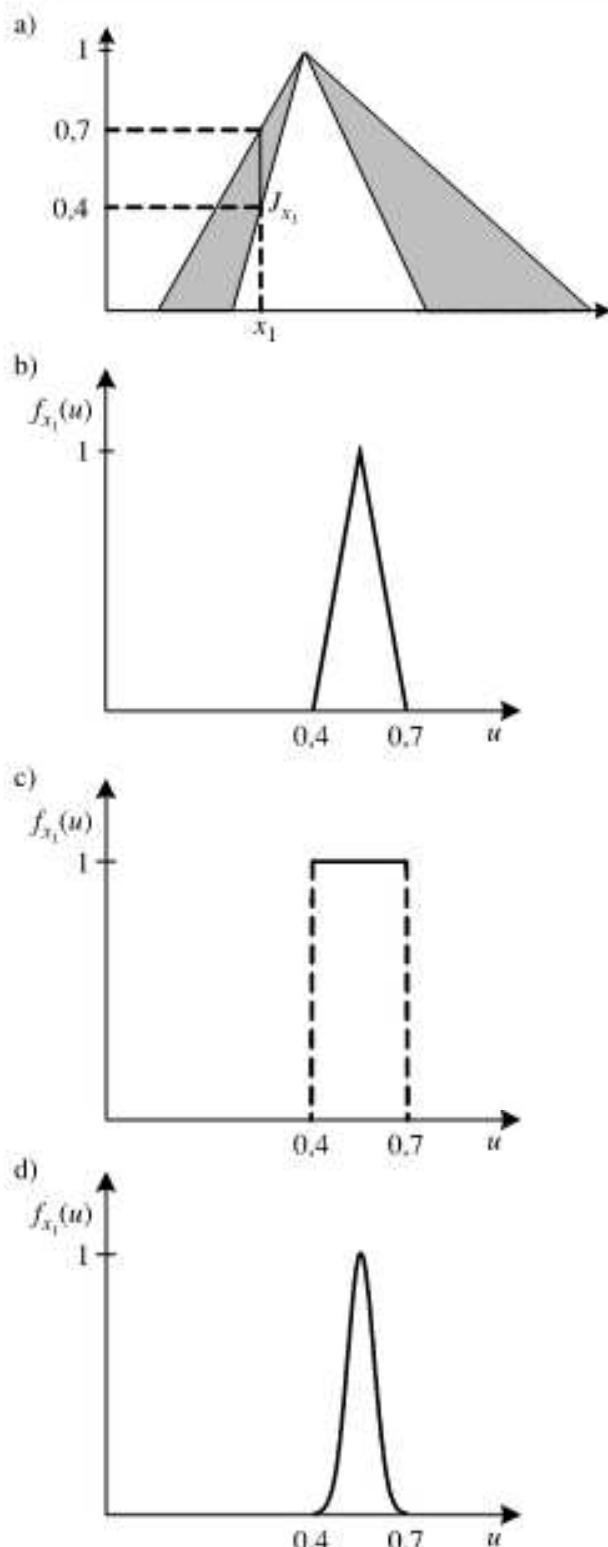
$$\tilde{A} = \int_{x \in X} \mu_{\tilde{A}}(x)/x \quad (5.3)$$

lub

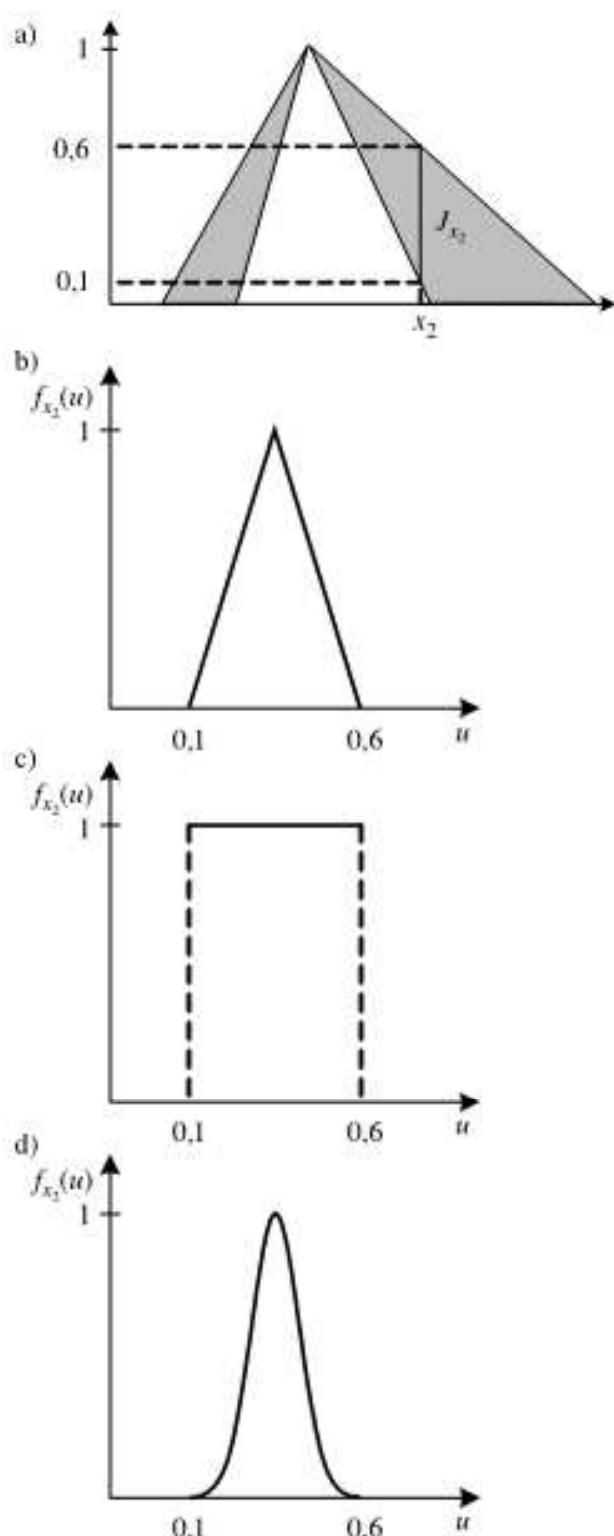
$$\tilde{A} = \int \mu_{\tilde{A}}(x)/x = \int_{x \in X} \left[\int_{u \in J_x} f_x(u)/u \right] / x, \quad J_x \subseteq [0, 1]. \quad (5.4)$$

Przykład 5.1

Rysunek 5.2a ilustruje sposób konstrukcji zbioru rozmytego typu 2. Dla ustalonego elementu x_1 otrzymujemy przedział $J_{x_1} = [0,4; 0,7]$ będący dziedziną funkcji drugorzędnej przynależności f_{x_1} . Na rysunkach 5.2b, 5.2c, 5.2d pokazano drugorzędne funkcje przynależności typu trójkątnego, przedziałowego i gaussowskiego o skończonym nośniku [171]. Rysunek 5.3 ilustruje ten sam zbiór rozmyty typu 2, ale zaznaczono na nim inny element x_2 oraz odpowiadający mu stopień przynależności będący zbiorem rozmytym typu 1



Rys. 5.2. Ilustracja zbioru rozmytego typu 2 wraz z drugorzędnymi funkcjami przynależności dla $J_{x_1} = [0,4; 0,7]$



Rys. 5.3. Ilustracja zbioru rozmytego typu 2 wraz z drugorzędnymi funkcjami przynależności dla $J_{x_2} = [0,1; 0,6]$

(trójkątnym, przedziałowym lub gaussowskim o skończonym nośniku) zdefiniowanym na przedziale $J_{x_2} = [0,1; 0,6]$.

W przypadku dyskretnym zbiór rozmyty typu 2 definiujemy analogicznie, tzn.

$$\tilde{A} = \sum_{x \in X} \mu_{\tilde{A}}(x)/x \quad (5.5)$$

oraz

$$\mu_{\tilde{A}}(x) = \sum_{u \in J_x} f_x(u)/u. \quad (5.6)$$

Założymy, że zbiór X został zdyskretyzowany i przyjmuje R wartości x_1, \dots, x_R , natomiast przedziały J_x odpowiadające tym wartościom zostały zdyskretyzowane i każdy z nich przyjmuje M_i wartości, $i = 1, \dots, R$. Wówczas możemy zapisać

$$\begin{aligned} \tilde{A} &= \sum_{x \in X} \left[\sum_{u \in J_x} f_x(u)/u \right] / x = \sum_{i=1}^R \left[\sum_{u \in J_{x_i}} f_{x_i}(u)/u \right] / x_i \\ &= \left[\sum_{k=1}^{M_1} f_{x_1}(u_{1k})/u_{1k} \right] / x_1 + \dots + \left[\sum_{k=1}^{M_R} f_{x_R}(u_{Rk})/u_{Rk} \right] / x_R. \end{aligned} \quad (5.7)$$

Uwaga 5.1

Rozmyty stopień przynależności może przybierać dwie charakterystyczne, skrajne postaci zbioru rozmytego typu 1:

$\mu_{\tilde{A}}(x) = 1/1$ oznaczającą pełną przynależność elementu x do zbioru rozmytego \tilde{A} ,
 $\mu_{\tilde{A}}(x) = 1/0$ oznaczającą brak przynależności elementu x do zbioru rozmytego \tilde{A} .

Przykład 5.2

Założymy, że $X = \{1, 2, 3\}$ oraz $J_{x_1} = \{0,2; 0,5; 0,7\}$, $J_{x_2} = \{0,5; 1\}$, $J_{x_3} = \{0,1; 0,3; 0,5\}$. Przypisując poszczególnym elementom zbiorów $J_{x_1}, J_{x_2}, J_{x_3}$ odpowiednie stopnie drugorzędnej przynależności, możemy zdefiniować następujący zbiór rozmyty typu 2:

$$\begin{aligned} \tilde{A} &= (0,5/0,2 + 1/0,5 + 0,5/0,7)/1 + (0,5/0,5 + 1/1)/2 \\ &\quad + (0,5/0,1 + 1/0,3 + 0,5/0,5)/3. \end{aligned} \quad (5.8)$$

DEFINICJA 5.2

Założymy, że każda funkcja drugorzędnej przynależności f_x zbioru rozmytego typu 2 przyjmuje wartość 1 tylko dla jednego elementu $u \in J_x$. Wówczas suma mnogościowa elementów u tworzy tzw. *funkcję głównej przynależności*, tzn.

$$\mu_{A_g}(x) = \int_{x \in X} u/x, \quad \text{gdzie } f_x(u) = 1. \quad (5.9)$$

Funkcja głównej przynależności definiuje odpowiedni zbiór rozmyty typu 1 oznaczony przez A_g .

Uwaga 5.2

W przypadku gdy funkcja drugorzędnej przynależności f_x jest funkcją przedziałową, wówczas funkcję głównej przynależności wyznaczamy jako sumę mnogościową wszystkich punktów u będących środkami podstawowej przynależności J_x , $x \in X$.

Przykład 5.3

Rozważmy zbiór rozmyty typu 2 dany wzorem (5.8). Na podstawie definicji 5.2 możemy określić następujący zbiór rozmyty A_g :

$$A_g = \frac{0,5}{1} + \frac{1}{2} + \frac{0,3}{3}. \quad (5.10)$$

5.3. Ślad niepewności

Zbiór rozmyty typu 2 można opisać, wykorzystując pojęcie śladu niepewności.

DEFINICJA 5.3

Niech $J_x \subset [0, 1]$ oznacza podstawową przynależność elementu x . *Śladem niepewności* SN zbioru rozmytego typu 2, $\tilde{A} \subseteq X$, nazywamy ograniczony obszar złożony ze wszystkich punktów podstawowej przynależności elementów x , tzn.

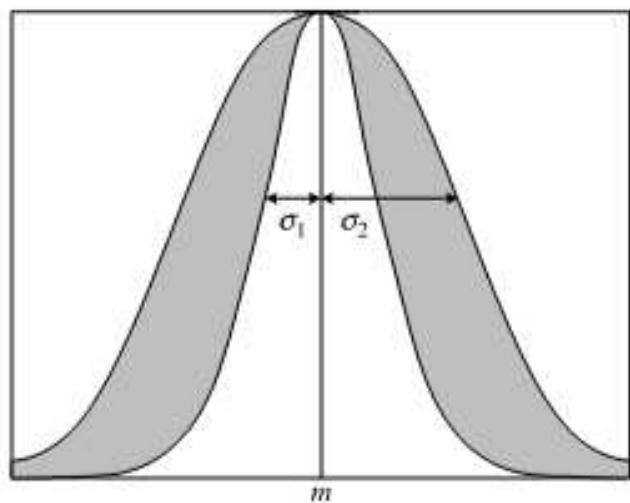
$$SN(\tilde{A}) = \bigcup_{x \in X} J_x. \quad (5.11)$$

Przykład 5.4

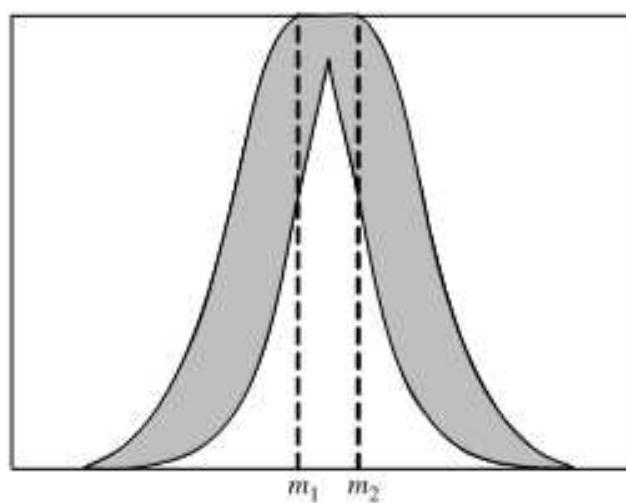
Rozważmy rodzinę funkcji przynależności zbioru rozmytego typu 1, który jest opisany funkcją gaussowską przy założeniu, że odchylenie standardowe σ zmienia się w przedziale $[\sigma_1, \sigma_2]$, tzn.

$$\mu_A(x) = N(m, \sigma; x) = \exp \left[-\frac{1}{2} \left(\frac{x - m}{\sigma} \right)^2 \right], \quad \sigma \in [\sigma_1, \sigma_2]. \quad (5.12)$$

Rodzina funkcji przynależności (5.12) tworzy zbiór rozmyty typu 2. Pełny opis tego zbioru wymagałby zdefiniowania drugorzędnej funkcji przynależności dla każdego punktu x i odpowiadającego mu przedziału J_x . Na rysunku 5.4 przedstawiono ślad niepewności rozważanego zbioru rozmytego typu 2.



Rys. 5.4. Ślad niepewności zbioru rozmytego typu 2: $\sigma \in [\sigma_1, \sigma_2]$



Rys. 5.5. Ślad niepewności zbioru rozmytego typu 2: $m \in [m_1, m_2]$

Przykład 5.5

Rozważmy rodzinę funkcji przynależności zbioru rozmytego typu 1, który jest opisany funkcją gaussowską przy założeniu, że wartość średnia m zmienia się w przedziale $[m_1, m_2]$, tzn.

$$\mu_A(x) = \exp \left[-\frac{1}{2} \left(\frac{x - m}{\sigma} \right)^2 \right], \quad m \in [m_1, m_2]. \quad (5.13)$$

Rodzina funkcji przynależności (5.13) tworzy zbiór rozmyty typu 2. Podobnie jak w poprzednim przykładzie, pełny opis tego zbioru wymagałby zdefiniowania drugorzędnnej funkcji przynależności dla każdego punktu x i odpowiadającego mu przedziału J_x . Na rysunku 5.5 przedstawiono ślad niepewności rozważanego zbioru rozmytego typu 2.

Założmy, że $J_x = [\underline{J}_x, \bar{J}_x]$, $x \in X$.

DEFINICJA 5.4

Funkcją górnej przynależności (FGP) nazywamy funkcję przynależności zbioru rozmytego typu 1 wyznaczoną poprzez

$$\bar{\mu}_{\tilde{A}}(x) = FGP(\tilde{A}) = \bigcup_{x \in X} \bar{J}_x \quad \forall x \in X. \quad (5.14)$$

DEFINICJA 5.5

Funkcją dolnej przynależności (FDP) nazywamy funkcję przynależności zbioru rozmytego typu 1 wyznaczoną poprzez

$$\underline{\mu}_{\tilde{A}}(x) = FDP(\tilde{A}) = \bigcup_{x \in X} \underline{J}_x \quad \forall x \in X. \quad (5.15)$$

Przykład 5.6

Wyznaczmy ślad niepewności dla zbioru rozmytego typu 2 określonego w przykładzie 5.4. Łatwo zauważyc, że funkcja górnej przynależności przybiera postać

$$\bar{\mu}_{\tilde{A}}(x) = N(m, \sigma_2; x), \quad (5.16)$$

natomiast funkcja dolnej przynależności jest dana wzorem

$$\underline{\mu}_{\tilde{A}}(x) = N(m, \sigma_1; x). \quad (5.17)$$

Przykład 5.7

Dla zbioru rozmytego typu 2 danego w przykładzie 5.5 funkcja górnej przynależności jest dana wzorem

$$\bar{\mu}_{\tilde{A}}(x) = \begin{cases} N(m_1, \sigma; x) & \text{dla } x < m_1, \\ 1 & \text{dla } m_1 \leq x \leq m_2, \\ N(m_2, \sigma; x) & \text{dla } x > m_2, \end{cases} \quad (5.18)$$

natomiast funkcja dolnej przynależności przybiera postać

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} N(m_2, \sigma; x) & \text{dla } x \leq \frac{m_1 + m_2}{2}, \\ N(m_1, \sigma; x) & \text{dla } x > \frac{m_1 + m_2}{2}. \end{cases} \quad (5.19)$$

5.4. Osadzone zbiory rozmyte

W zbiorach rozmytych typu 2 możemy wyróżnić tzw. osadzone zbiory rozmyte typu 1 i typu 2.

DEFINICJA 5.6

Wybierzmy z każdego przedziału J_x , $x \in X$, tylko jeden element $\theta \in J_x$.

Zbiorem osadzonym typu 2 w zbiorze \tilde{A} jest zbiór \tilde{A}_o

$$\tilde{A}_o = \int_{x \in X} [f_x(\theta)/\theta]/x \quad \theta \in J_x \subseteq U = [0, 1]. \quad (5.20)$$

Oczywiście, istnieje nieprzeliczalna liczba zbiorów osadzonych \tilde{A}_o w zbiorze \tilde{A} . W przypadku dyskretnym zbiór osadzony \tilde{A}_o jest zdefiniowany następująco:

$$\tilde{A}_o = \sum_{i=1}^R [f_{x_i}(\theta_i)/\theta_i]/x_i \quad \theta_i \in J_{x_i} \subseteq U = [0, 1]. \quad (5.21)$$

Latwo zauważyć, że istnieje $\prod_{i=1}^R M_i$ osadzonych zbiorów rozmytych \tilde{A}_o w zbiorze \tilde{A} .

Przykład 5.8

Założymy, że

$$\begin{aligned} \tilde{A} = & (0.5/0.2 + 1/0.5 + 0.5/0.7)/2 + (0.3/0.5 + 1/1)/3 \\ & + (0.5/0.1 + 1/0.3 + 0.5/0.5)/4. \end{aligned} \quad (5.22)$$

Wówczas jeden z 18 osadzonych zbiorów rozmytych \tilde{A}_o jest postaci

$$\tilde{A}_o = (0.5/0.7)/2 + (0.3/0.5)/3 + (1/0.3)/4. \quad (5.23)$$

Z każdym osadzonym zbiorem rozmytym typu 2, \tilde{A}_o , związany jest osadzony zbiór rozmyty typu 1 oznaczony jako A_o .

DEFINICJA 5.7

Zbiór osadzony typu 1 definiujemy następująco:

$$A_o = \int_{x \in X} \theta/x \quad \theta \in J_x \subseteq U = [0, 1]. \quad (5.24)$$

Istnieje nieprzeliczalna liczba osadzonych zbiorów rozmytych A_o . W przypadku dyskretnym wzór (5.24) przybiera postać

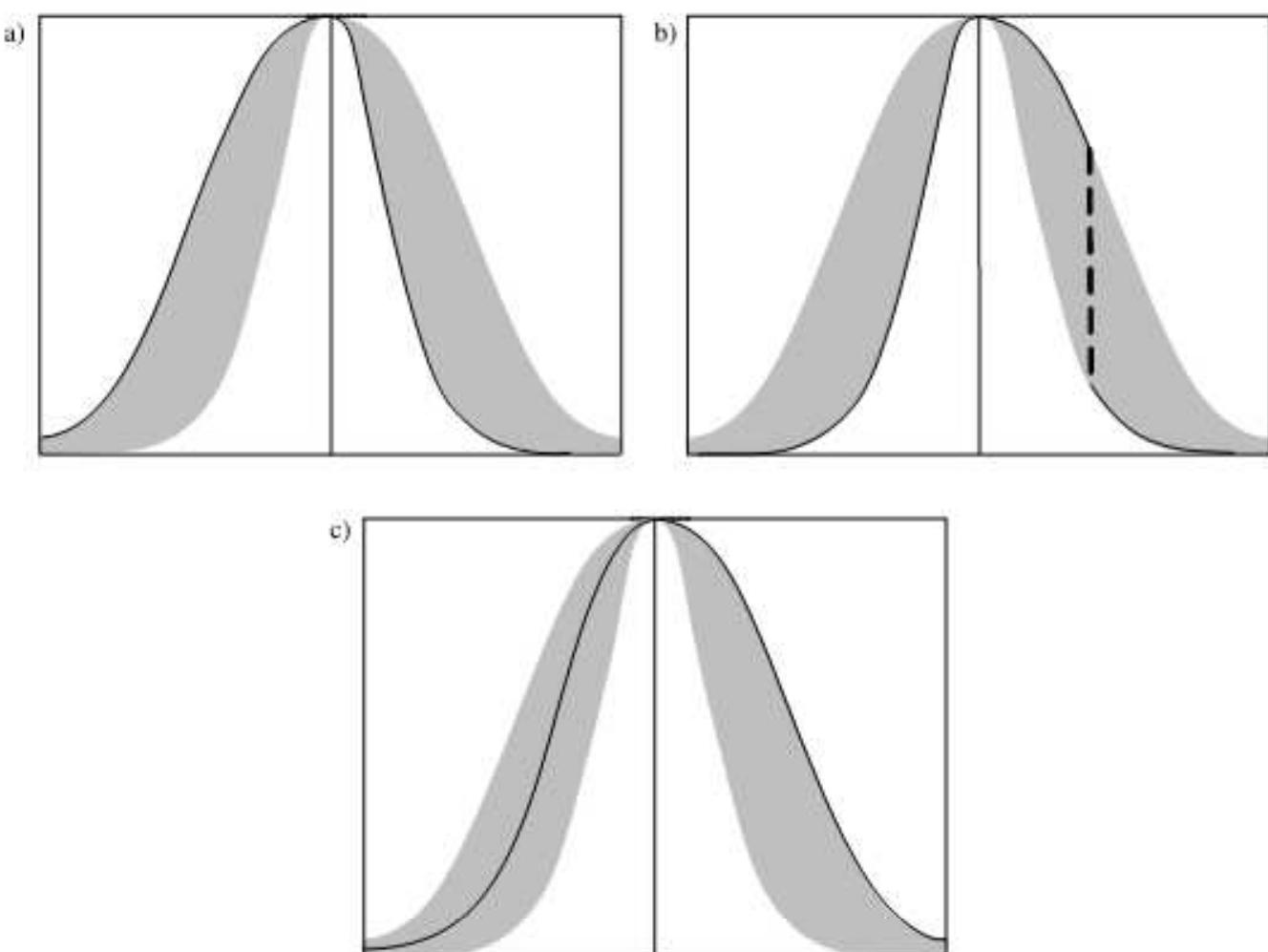
$$A_o = \sum_{i=1}^R \theta_i/x_i \quad \theta_i \in J_{x_i} \subseteq U = [0, 1]. \quad (5.25)$$

Liczba wszystkich zbiorów A_o wynosi $\prod_{i=1}^R M_i$.

Szczególnym przypadkiem zbioru osadzonego typu 1 jest zbiór rozmyty A_g zdefiniowany poprzez funkcję głównej przynależności daną wzorem (5.9). Ponadto należy zauważyć, że osadzony zbiór rozmyty A_o gubi wszystkie informacje o stopniach drugorzędnej przynależności. Zatem na podstawie rodziny zbiorów osadzonych A_o nie można odtworzyć zbioru rozmytego typu 2, a jedynie jego ślad niepewności. Jednak pojęcie zbioru osadzonego okaże się niezwykle przydatne przy omawianiu szybkiego algorytmu redukcji typu przedstawionego w dalszej części rozdziału.

Przykład 5.9

Na rysunku 5.6 przedstawiono trzy różne zbiory osadzone typu 1.



Rys. 5.6. Zbiory osadzone typu 1

Przykład 5.10

Rozważmy zbiór rozmyty typu 2 postaci

$$\tilde{A} = (0,6/0,3 + 1/0,7)/3 + (0,4/0,4 + 1/1)/5. \quad (5.26)$$

W zbiorze \tilde{A} możemy wyróżnić cztery osadzone zbiory rozmyte A_o :

$$\begin{aligned} A_o &= 0,3/3 + 0,4/5, \\ A_o &= 0,7/3 + 0,4/5, \\ A_o &= 0,3/3 + 1/5, \\ A_o &= 0,7/3 + 1/5. \end{aligned} \quad (5.27)$$

5.5. Podstawowe operacje na zbiorach rozmytych typu 2

Zasada rozszerzania (podrozdz. 4.4) pozwala rozszerzyć operacje na zbiorach rozmytych typu 1 do działań na zbiorach typu 2.

Rozważmy dwa zbiory rozmyte typu 2, \tilde{A} i \tilde{B} , zdefiniowane następująco:

$$\tilde{A} = \int_{x \in X} \left(\int_{u \in J_x^u} f_x(u)/u \right) / x \quad (5.28)$$

oraz

$$\tilde{B} = \int_{x \in X} \left(\int_{v \in J_x^v} g_x(v)/v \right) / x, \quad (5.29)$$

gdzie $J_x^u \cdot J_x^v \subset [0, 1]$. Sumę zbiorów \tilde{A} i \tilde{B} stanowi zbiór rozmyty typu 2, który oznaczamy jako $\tilde{A} \cup \tilde{B}$ i określamy następująco:

$$\begin{aligned} \mu_{\tilde{A} \cup \tilde{B}}(x) &= \int_{w \in J_x^w} h_x(w)/w = \phi(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)) \\ &= \phi\left(\int_{u \in J_x^u} f_x(u)/u, \int_{v \in J_x^v} g_x(v)/v\right). \end{aligned} \quad (5.30)$$

W tym przypadku rozszerzaną funkcją ϕ jest dowolna t -konorma, natomiast jej argumentami nie są już zwykłe liczby, ale zbiory rozmyte typu 1 $\mu_{\tilde{A}}(x)$ i $\mu_{\tilde{B}}(x)$ dla ustalonego $x \in X$. Zgodnie z zasadą rozszerzania

$$\phi\left(\int_{u \in J_x^u} f_x(u)/u, \int_{v \in J_x^v} g_x(v)/v\right) = \int_{u \in J_x^u} \int_{v \in J_x^v} f_x(u)^T * g_x(v)/\phi(u, v). \quad (5.31)$$

Po podstawieniu t -konormy w miejsce funkcji ϕ , suma zbiorów rozmytych typu 2 jest określona rozmytą funkcją przynależności

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = \int_{u \in J_x^u} \int_{v \in J_x^v} f_x(u)^T * g_x(v)/u^S * v. \quad (5.32)$$

Powyższa formuła pozwala na wyznaczenie sumy zbiorów rozmytych typu 2 dla każdej wartości x . Funkcja przynależności zbioru wynikowego jest największą wartością wyrażenia $f_x(u)^T * g_x(v)$ po wszystkich parach (u, v) , które dają w wyniku ten sam element $w = u^S * v$.

Przykład 5.11

W tym przykładzie wyjaśnimy szczegółowo sposób wyznaczania sumy zbiorów rozmytych typu 2. Jako t -normę przyjmujemy operację minimum, a jako t -konormę operację maksimum. Rozważmy dwa zbiory rozmyte typu 2, \tilde{A} i \tilde{B} , takie, że

$$\begin{aligned} \tilde{A} &= (0,5/0,2 + 1/0,5 + 0,5/0,7)/1 + (0,5/0,5 + 1/1)/2 \\ &\quad + (0,5/0,1 + 1/0,3 + 0,5/0,5)/3 \end{aligned} \quad (5.33)$$

oraz

$$\tilde{B} = (1/0)/1 + (0,5/0,5 + 1/0,8)/2 + (1/0,6 + 0,5/1)/3. \quad (5.34)$$

Zgodnie ze wzorem (5.32) dla $x = 1$ mamy

$$\mu_{\tilde{A} \cup \tilde{B}}(1) = \frac{0,5 \wedge 1}{0,2 \vee 0} + \frac{1 \wedge 1}{0,5 \vee 0} + \frac{0,5 \wedge 1}{0,7 \vee 0} = \frac{0,5}{0,2} + \frac{1}{0,5} + \frac{0,5}{0,7}. \quad (5.35)$$

Dla $x = 2$ otrzymujemy

$$\begin{aligned}\mu_{\tilde{A} \cup \tilde{B}}(2) &= \frac{0,5 \wedge 0,5}{0,5 \vee 0,5} + \frac{0,5 \wedge 1}{0,5 \vee 0,8} + \frac{\max(1 \wedge 0,5; 1 \wedge 1)}{1} \\ &= \frac{0,5}{0,5} + \frac{0,5}{0,8} + \frac{1}{1}.\end{aligned}\quad (5.36)$$

Dla $x = 3$ mamy

$$\begin{aligned}\mu_{\tilde{A} \cup \tilde{B}}(3) &= \frac{\max(0,5 \wedge 1; 1 \wedge 1; 0,5 \wedge 1)}{0,6} + \frac{\max(0,5 \wedge 0,5; 1 \wedge 0,5; 0,5 \wedge 0,5)}{1} \\ &= \frac{1}{0,6} + \frac{0,5}{1}.\end{aligned}\quad (5.37)$$

Zatem suma zbiorów \tilde{A} i \tilde{B} wynosi

$$\begin{aligned}\tilde{A} \cup \tilde{B} &= (0,5/0,2 + 1/0,5 + 0,5/0,7)/1 + (0,5/0,5 + 0,5/0,8 + 1/1)/2 \\ &\quad + (1/0,6 + 0,5/1)/3.\end{aligned}\quad (5.38)$$

Przecięcie zbiorów \tilde{A} i \tilde{B} jest zbiorem rozmytym typu 2 o rozmytej funkcji przynależności danej wzorem

$$\begin{aligned}\mu_{\tilde{A} \cap \tilde{B}}(x) &= \int_{w \in J_x^*} h_x(w)/w = \phi(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)) \\ &= \phi\left(\int_{u \in J_x^u} f_x(u)/u, \int_{v \in J_x^v} g_x(v)/v\right),\end{aligned}\quad (5.39)$$

przy czym rozszerzaną funkcją ϕ jest tym razem dowolna t -norma. Niezmiennie argumentami funkcji ϕ są zbiory rozmyte typu 1, tzn. $\mu_{\tilde{A}}(x)$ i $\mu_{\tilde{B}}(x)$. Zatem przecięcie zbiorów rozmytych typu 2 określone jest następująco:

$$\mu_{\tilde{A} \cap \tilde{B}}(x) = \int_{u \in J_x^u} \int_{v \in J_x^v} f_x(u) \stackrel{T^*}{*} g_x(v) / u \stackrel{T}{*} v. \quad (5.40)$$

We wzorze (5.40) t -norma agregująca drugorzędne przynależności oznaczona została przez T^* , a jej postać funkcyjna może być dobrana niezależnie od doboru rozszerzanej t -normy T . Także w tym przypadku funkcja przynależności zbioru wynikowego jest największą wartością wyrażenia $f_x(u) \stackrel{T^*}{*} g_x(v)$ po wszystkich parach (u, v) , które dają w wyniku ten sam element $w = u \stackrel{T}{*} v$.

Przykład 5.12

Wyznaczmy przecięcie zbiorów rozmytych typu 2 rozważanych w przykładzie 5.11. Jako t -normę T^* oraz T przyjmujemy operację minimum. Zgodnie ze wzorem (5.40) dla $x = 1$ oraz $x = 2$ otrzymujemy

$$\mu_{\tilde{A} \cap \tilde{B}}(1) = \frac{\max(0,5 \wedge 1; 1 \wedge 1; 0,5 \wedge 1)}{0} = \frac{1}{0} \quad (5.41)$$

oraz

$$\mu_{\tilde{A} \cap \tilde{B}}(2) = \frac{\max(0,5 \wedge 0,5; 0,5 \wedge 1; 1 \wedge 0,5)}{0,5} + \frac{1 \wedge 1}{1 \wedge 0,8} = \frac{0,5}{0,5} + \frac{1}{0,8}. \quad (5.42)$$

Dopełnienie zbioru rozmytego typu 2 jest zbiorem rozmytym typu 2 o rozmytej funkcji przynależności danej wzorem

$$\begin{aligned}\mu_{\tilde{A}}^c(x) &= \phi(\mu_{\tilde{A}}(x)) \\ &= \int_{u \in J_x^u} f_x(u)/(1-u).\end{aligned}\quad (5.43)$$

Przykład 5.13

Rozważmy zbiór rozmyty typu 2 dany wzorem

$$\mu_{\tilde{A}}(x) = (0,4/0,6 + 1/0,7)/9. \quad (5.44)$$

Zgodnie ze wzorem (5.43) mamy

$$\mu_{\tilde{A}}^c(x) = (0,4/0,4 + 1/0,3)/9. \quad (5.45)$$

Uwaga 5.3

Sumę (5.32) oraz przecięcie (5.40) zbiorów rozmytych typu 2 można traktować jako wynik zastosowania operatora rozszerzonej t -normy \tilde{T} oraz rozszerzonej t -konormy \tilde{S} . Operatory te można rozważać także w kontekście zbiorów rozmytych typu 1. Rozważmy dwa takie zbiory

$$F = \int_{u \in J^u} f(u)/u \quad \text{oraz} \quad G = \int_{v \in J^v} g(v)/v. \quad (5.46)$$

Operator rozszerzonej t -normy, której argumentami i wartością wynikową są zbiory rozmyte typu 1 zdefiniowane w obszarze rozważań $[0, 1]$, jest postaci

$$F \tilde{*} G = \int_{u \in J^u} \int_{v \in J^v} g(u) \stackrel{T^*}{*} f(v)/u \stackrel{T}{*} v. \quad (5.47)$$

Analogiczny rezultat uzyskujemy w przypadku dyskretnym. Rozważmy dwa zbiory typu 1

$$F = \sum_{u \in J^u} f(u)/u \quad \text{oraz} \quad G = \sum_{v \in J^v} g(v)/v. \quad (5.48)$$

Operator rozszerzonej t -normy dany jest wzorem

$$\mu_{\tilde{A}} \tilde{*} \mu_{\tilde{B}} = \sum_{u \in J^u} \sum_{v \in J^v} (f(u) \stackrel{T^*}{*} g(v))/u \stackrel{T}{*} v, \quad (5.49)$$

natomiast operator rozszerzonej t -konormy określamy następująco:

$$\mu_{\tilde{A}} \tilde{*} \mu_{\tilde{B}} = \sum_{u \in J^u} \sum_{v \in J^v} (f(u) \stackrel{T}{*} g(v))/u \stackrel{S}{*} v. \quad (5.50)$$

Wprowadzenie operacji rozszerzonych norm na zbiorach typu 1 pozwala na znaczne uproszczenie zapisu zawiłych operacji na zbiorach rozmytych typu 2.

Uwaga 5.4

Rozszerzana funkcja ϕ może być również funkcją wielu zmiennych. Wówczas operacje rozszerzonej t -normy i t -konormy przybierają następującą postać wieloargumentową:

$$\tilde{\tilde{T}}_{i=1}^n F_i = \int_{u_1 \in J_1} \dots \int_{u_n \in J_n} \tilde{T}^* f_i(u_i) / \tilde{T} \prod_{i=1}^n u_i, \quad (5.51)$$

$$\tilde{\tilde{S}}_{i=1}^n F_i = \int_{u_1 \in J_1} \dots \int_{u_n \in J_n} \tilde{T} f_i(u_i) / \tilde{S} \prod_{i=1}^n u_i, \quad (5.52)$$

gdzie $F_i = \int_{u_i \in J_i} f_i(u_i) / u_i$, $i = 1, \dots, n$.

Uwaga 5.5

Operacje rozszerzonej t -normy i t -konormy są łatwiejsze do realizacji przy określonych założeniach odnośnie do funkcji przynależności poszczególnych zbiorów rozmytych. Rozważmy n wypukłych, normalnych zbiorów rozmytych (typu 1) F_1, \dots, F_n z funkcjami przynależności f_1, \dots, f_n . Założymy, że $f_1(v_1) = f_2(v_2) = \dots = f_n(v_n) = 1$, gdzie v_1, v_2, \dots, v_n są liczbami rzeczywistymi takimi, że $v_1 \leq v_2 \leq \dots \leq v_n$. Wówczas rozszerzona t -norma typu minimum jest określona następująco ([97, 134]):

$$\mu_{\cap_{i=1}^n F_i}(\theta) = \begin{cases} \bigvee_{i=1}^n f_i(\theta), & \theta < v_1, \\ \bigwedge_{i=1}^k f_i(\theta), & v_k \leq \theta < v_{k+1}, \quad 1 \leq k \leq n-1, \\ \bigwedge_{i=1}^n f_i(\theta), & \theta \geq v_n, \end{cases} \quad (5.53)$$

natomiast rozszerzona t -konorma typu maksimum przybiera postać

$$\mu_{\cup_{i=1}^n F_i}(\theta) = \begin{cases} \bigwedge_{i=1}^n f_i(\theta), & \theta < v_1, \\ \bigwedge_{i=k+1}^n f_i(\theta), & v_k \leq \theta < v_{k+1}, \quad 1 \leq k \leq n-1, \\ \bigvee_{i=1}^n f_i(\theta), & \theta \geq v_n. \end{cases} \quad (5.54)$$

Uwaga 5.6

Rozważmy n gaussowskich zbiorów rozmytych F_1, F_2, \dots, F_n ze średnimi m_1, m_2, \dots, m_n oraz odchyleniami standardowymi $\sigma_1, \sigma_2, \dots, \sigma_n$. Wówczas w wyniku rozszerzonej operacji t -normy algebraicznej otrzymujemy [97]

$$\mu_{F_1 \cap F_2 \cap \dots \cap F_n}(\theta) \approx e^{(1/2)((\theta - m_1 m_2 \dots m_n) / \bar{\sigma})^2}, \quad (5.55)$$

przy czym

$$\bar{\sigma} = \sqrt{\sigma_1^2 \prod_{i:i \neq 1} m_i^2 + \dots + \sigma_j^2 \prod_{i:i \neq j} m_i^2 + \dots + \sigma_n^2 \prod_{i:i \neq 1} m_i^2}, \quad (5.56)$$

gdzie $i = 1, \dots, n$.

5.6. Relacje rozmyte typu 2

Najpierw zdefiniujemy iloczyn kartezjański zbiorów rozmytych typu 2.

DEFINICJA 5.8

Iloczynem kartezjańskim n zbiorów rozmytych typu 2 $\tilde{A}_1 \subseteq X_1, \tilde{A}_2 \subseteq X_2, \dots, \tilde{A}_n \subseteq X_n$ jest zbiór rozmyty $\tilde{A} = \tilde{A}_1 \times \tilde{A}_2 \times \dots \times \tilde{A}_n$, zdefiniowany na zbiorze $X_1 \times X_2 \times \dots \times X_n$, przy czym funkcja przynależności zbioru \tilde{A} jest dana wzorem

$$\mu_{\tilde{A}}(\mathbf{x}) = \mu_{\tilde{A}_1 \times \tilde{A}_2 \times \dots \times \tilde{A}_n}(x_1, x_2, \dots, x_n) = \prod_{i=1}^n \mu_{\tilde{A}_i}(x_i), \quad (5.57)$$

gdzie $x_1 \in X_1, \dots, x_n \in X_n$, natomiast operacja rozszerzonej t -normy opisana jest zależnością (5.51).

DEFINICJA 5.9

Binarną relacją rozmytą typu 2 \tilde{R} między dwoma niepustymi zbiorami nierożmytymi X i Y nazywamy zbiór rozmyty typu 2 określony na iloczynie kartezjańskim $X \times Y$, tzn.

$$\tilde{R}(X, Y) = \int_{X \times Y} \mu_{\tilde{R}}(x, y) / (x, y), \quad (5.58)$$

przy czym $x \in X, y \in Y$, natomiast stopień przynależności pary (x, y) do zbioru rozmytego \tilde{R} jest zbiorem rozmytym typu 1 określonym na przedziale $J_{x,y}^v \subset [0, 1]$, tzn.

$$\mu_{\tilde{R}}(x, y) = \int_{v \in J_{x,y}^v} r_{x,y}(v) / v, \quad (5.59)$$

gdzie $r_{x,y}(v)$ jest stopniem drugorzędnej przynależności.

Przykład 5.13

Niech $X = \{3, 4\}$ oraz $Y = \{4, 5\}$. Sformalizujmy nieprecyzyjne stwierdzenie „ y jest mniej więcej równe x ”. Najpierw określmy relacje n typu 1 w sposób następujący:

$$R = \frac{0,8}{(3, 4)} + \frac{0,6}{(3, 5)} + \frac{1}{(4, 4)} + \frac{0,8}{(4, 5)}. \quad (5.60)$$

Analogiczna relacja rozmyta typu 2 może przybrać formę

$$\begin{aligned} \tilde{R} = & (0,6/0,7 + 1/0,8 + 0,5/0,6) / (3; 4) + (0,3/0,5 + 1/0,6 + 0,4/0,3) / (3; 5) \\ & + (1/1 + 1/1 + 1/1) / (4; 4) + (0,6/0,7 + 1/0,8 + 0,5/0,6) / (4; 5). \end{aligned} \quad (5.61)$$

Przykład 5.14

Sformalizujmy nieprecyzyjne stwierdzenie „liczba x niewiele się różni od liczby y ”. Zadanie to możemy rozwiązać za pomocą relacji rozmytej typu 1, opisanej funkcją przynależności

$$\mu_R(x, y) = \max\{(4 - |x - y|) / 4, 0\}. \quad (5.62)$$

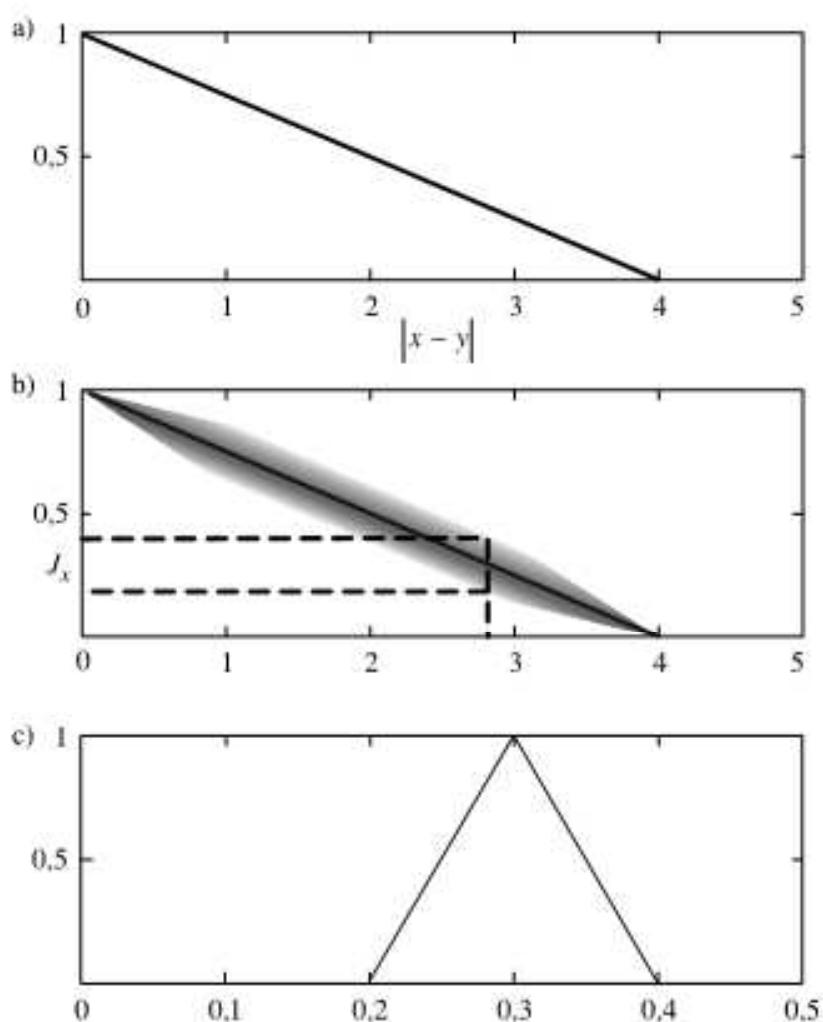
Analogiczna relacja rozmyta typu 2 może przybrać postać

$$\mu_{\tilde{R}}(x, y) = \int_{v \in [0,1]} \exp\left[-\left(\frac{v - m(x, y)}{\sigma}\right)^2\right] / v, \quad (5.63)$$

gdzie $\sigma > 0$ oraz

$$m(x, y) = \max\{(4 - |x - y|)/4, 0\}. \quad (5.64)$$

Alternatywnie, drugorzędna funkcja przynależności typu gaussowskiego może być zastąpiona rozmytą liczbą trójkątną. Na rysunku 5.7a przedstawiono ilustrację relacji rozmytej typu 1 danej wzorem (5.62). Rysunek 5.7b odzwierciedla możliwość niepewności w sprecyzowaniu pojęcia „liczba x niewiele się różni od liczby y ”. Na rysunku tym uwidoczniono ślad niepewności, przy czym stopień zaciemnienia odpowiada wartości stopnia drugorzędnej przynależności. Rysunek 5.7c przedstawia trójkątną funkcję drugorzędnej przynależności zdefiniowaną na odcinku $J_x[0, 2; 0, 4]$.



Rys. 5.7. Ilustracja relacji rozmytej typu 2

Warto wspomnieć, że relacje rozmyte mogą być realizowane z użyciem rozszerzonych norm komplementarnych. Rozważmy funkcję przynależności zbioru rozmytego typu 2 określonego na zbiorze X , $\tilde{A} \subseteq X$, tzn.

$$\mu_{\tilde{A}}(x) = \int_{u \in J_x^u} f_x(u)/u \quad (5.65)$$

oraz funkcję przynależności zbioru rozmytego \tilde{B} określona na innym zbiorze Y , $\tilde{B} \subset Y$, tzn.

$$\mu_{\tilde{B}}(y) = \int_{v \in J_y^v} g_y(v)/v, \quad (5.66)$$

gdzie $J_x^u, J_y^v \subset [0, 1]$. Rozszerzona t -konorma zbiorów rozmytych typu 2, określonych na różnych przestrzeniach, tworzy pewną relację rozmytą \tilde{R} , określoną w sposób następujący:

$$\begin{aligned}\mu_{\tilde{R}}(x, y) &= \mu_{\tilde{A}}(x) \tilde{*} \mu_{\tilde{B}}(y) = \int_{u \in J_x^u} \int_{v \in J_y^v} f_x(u) \tilde{*} g_y(v) / u \tilde{*} v \\ &= \int_{w \in J_{x,y}^w} r_{x,y}(w) / w.\end{aligned}\quad (5.67)$$

Podobnie rozszerzona t -norma tworzy relację rozmytą postaci

$$\begin{aligned}\mu_{\tilde{R}}(x, y) &= \mu_{\tilde{A}}(x) \tilde{*} \mu_{\tilde{B}}(y) = \int_{u \in J_x^u} \int_{v \in J_y^v} f_x(u) \tilde{*} g_y(v) / u \tilde{*} v \\ &= \int_{w \in J_{x,y}^w} r_{x,y}(w) / w.\end{aligned}\quad (5.68)$$

W zastosowaniu teorii zbiorów rozmytych do konstrukcji systemów wnioskujących niezbędne jest pojęcie złożenia dwóch relacji rozmytych, które w kontekście relacji rozmytych typu 2 definiowane są następująco.

DEFINICJA 5.10

Rozszerzonym złożeniem typu sup-T (sup-star) relacji rozmytych typu 2 $\tilde{R} \subseteq X \times Y$ i $\tilde{S} \subseteq Y \times Z$ nazywamy relację rozmytą $\tilde{R} \circ \tilde{S} \subseteq X \times Z$ o funkcji przynależności

$$\mu_{\tilde{R} \circ \tilde{S}}(\mathbf{x}, \mathbf{z}) = \tilde{S}_{y \in Y} (\mu_{\tilde{R}}(\mathbf{x}, \mathbf{y}) \tilde{*} \mu_{\tilde{S}}(\mathbf{y}, \mathbf{z})). \quad (5.69)$$

DEFINICJA 5.11

Rozszerzone złożenie zbioru rozmytego typu 2 \tilde{A} , $\tilde{A} \subset X$, oraz relacji rozmytej typu 2 $\tilde{R} \subseteq X \times Y$ zapisujemy jako $\tilde{A} \circ \tilde{R}$ i określamy następująco:

$$\mu_{\tilde{B}}(y) = \tilde{S}_{x \in X} (\mu_{\tilde{A}}(\mathbf{x}) \tilde{*} \mu_{\tilde{R}}(\mathbf{x}, y)). \quad (5.70)$$

5.7. Redukcja typu

Wyostrzanie zbiorów rozmytych typu 2 składa się z dwóch etapów. Najpierw należy dokonać tzw. *redukcji typu*, która polega na przekształceniu zbioru rozmytego typu 2 w zbiór rozmyty typu 1. Otrzymany w ten sposób zbiór rozmyty typu 1, nazywany *centroidem*, może być wyostrzony do wartości nierożmytej. Pokażemy teraz sposób wyznaczenia centroidu zbioru rozmytego typu 2.

Rozważmy zbiór rozmyty A (typu 1) określony na zbiorze X . Założymy, że zbiór X został zdyskretyzowany i przyjmuje R wartości x_1, \dots, x_R . Centroid zbioru rozmytego A jest określony następująco:

$$C_A = \frac{\sum_{k=1}^R x_k \mu_A(x_k)}{\sum_{k=1}^R \mu_A(x_k)}. \quad (5.71)$$

Wyznaczamy teraz centroid zbioru rozmytego typu 2, $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$, który w wyniku analogicznej dyskretyzacji zapisujemy następująco:

$$\tilde{A} = \sum_{k=1}^R \left[\int_{u \in J_{x_k}} f_{x_k}(u)/u \right] / x_k. \quad (5.72)$$

Stosując zasadę rozszerzania do wzoru (5.69), otrzymujemy

$$C_{\tilde{A}} = \int_{\theta_1 \in J_{x_1}} \dots \int_{\theta_R \in J_{x_R}} [f_{x_1}(\theta_1) * \dots * f_{x_R}(\theta_R)] / \frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k}. \quad (5.73)$$

Oczywiście centroid $C_{\tilde{A}}$ jest zbiorem rozmytym typu 1. Zauważmy, że dowolny wybór elementów $\theta_1 \in J_{x_1}, \dots, \theta_R \in J_{x_R}$ wraz z odpowiadającymi im stopniami drugorzędnej przynależności $f_{x_1}(\theta_1), \dots, f_{x_R}(\theta_R)$ tworzy osadzony zbiór rozmyty \tilde{A}_0 (typu 2).

Przykład 5.15

Niech $X = \{2, 5\}$. Dokonamy redukcji typu następującego zbioru rozmytego typu 2:

$$\tilde{A} = (0,6/0,4 + 1/0,8)/2 + (0,3/0,7 + 1/0,6)/5, \quad (5.74)$$

Centroid zbioru rozmytego typu 2 danego wzorem (5.74) jest zbiorem rozmytym typu 1 postaci

$$C_{\tilde{A}} = \frac{0,6 \times 0,3}{a_1} + \frac{0,6 \times 1}{a_2} + \frac{1 \times 0,3}{a_3} + \frac{1 \times 1}{a_4} = \frac{0,18}{a_1} + \frac{0,6}{a_2} + \frac{0,3}{a_3} + \frac{1}{a_4}, \quad (5.75)$$

przy czym

$$a_1 = \frac{2 \times 0,4 + 5 \times 0,7}{0,4 + 0,7} = \frac{43}{11},$$

$$a_2 = \frac{2 \times 0,4 + 5 \times 0,6}{0,4 + 0,6} = 3,8,$$

$$a_3 = \frac{2 \times 0,8 + 5 \times 0,7}{0,8 + 0,7} = 3,4,$$

$$a_4 = \frac{2 \times 0,8 + 5 \times 0,6}{0,8 + 0,6} = \frac{23}{7}.$$

W przypadku ciągłym wyznaczenie centroidu zbioru rozmytego typu 2 jest dużo bardziej skomplikowanym zadaniem z obliczeniowego punktu widzenia. Problem staje się łatwiejszy do rozwiązania, jeżeli funkcje drugorzędnej przynależności są przedziałowe. Wówczas wzór (5.73) przybiera postać

$$C_{\tilde{A}} = \int_{\theta_1 \in J_{x_1}} \dots \int_{\theta_R \in J_{x_R}} 1 / \frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k}. \quad (5.76)$$

Pokażemy teraz sposób wyznaczenia centroidu zbioru rozmytego typu 2 z przedziałową funkcją drugorzędnej przynależności. W nawiązaniu do wzoru (5.76) wprowadźmy oznaczenie

$$s(\theta_1, \dots, \theta_R) = \frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k}. \quad (5.77)$$

Jest oczywiste, że centroid (5.76) będzie przedziałowym zbiorem rozmytym typu 1, tzn.

$$C_{\tilde{A}} = \int_{x \in [x_l, x_p]} 1/x \equiv [x_l, x_p]. \quad (5.78)$$

Z powyższej obserwacji wynika, że wyznaczenie centroidu (5.76) sprowadza się do optymalizacji (maksymalizacji i minimalizacji) względem θ_k funkcji danej wzorem (5.77) z uwzględnieniem ograniczeń

$$\theta_k \in [\underline{\theta}^k, \bar{\theta}^k], \quad (5.79)$$

gdzie $k = 1, \dots, R$, oraz

$$\underline{\theta}^k = J_x, \bar{\theta}^k = \bar{J}_x. \quad (5.80)$$

Różniczkując wyrażenie (5.77) względem θ_j , otrzymujemy

$$\begin{aligned} \frac{\partial}{\partial \theta_j} s(\theta_1, \dots, \theta_R) &= \frac{\partial}{\partial \theta_j} \left[\frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k} \right] = \frac{\partial}{\partial \theta_j} \left[\frac{x_j \theta_j + \sum_{k \neq j} x_k \theta_k}{\theta_j + \sum_{k \neq j} \theta_k} \right] \\ &= \left[\frac{1}{\theta_j + \sum_{k \neq j} \theta_k} \right] (x_j) + \left(x_j \theta_j + \sum_{k \neq j} x_k \theta_k \right) \left[\frac{-1}{(\theta_j + \sum_{k \neq j} \theta_k)^2} \right] \\ &= \frac{x_j}{\sum_{k=1}^R \theta_k} - \frac{\sum_{k=1}^R x_k \theta_k}{(\sum_{k=1}^R \theta_k)^2} = \frac{x_j}{\sum_{k=1}^R \theta_k} - \left[\frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k} \right] \frac{1}{\sum_{k=1}^R \theta_k} \\ &= \frac{x_j - s(\theta_1, \dots, \theta_R)}{\sum_{k=1}^R \theta_k}. \end{aligned} \quad (5.81)$$

Oczywiście $\sum_{k=1}^R \theta_k > 0$. Zatem z ostatniej równości wynika, że

$$\frac{\partial}{\partial \theta_j} s(\theta_1, \dots, \theta_R) \geq 0, \quad \text{jeśli } x_j \geq s(\theta_1, \dots, \theta_R) \quad (5.82)$$

oraz

$$\frac{\partial}{\partial \theta_j} s(\theta_1, \dots, \theta_R) \leq 0, \quad \text{jeśli } x_j \leq s(\theta_1, \dots, \theta_R). \quad (5.83)$$

Przyrównując do zera prawą stronę wyrażenia (5.81), mamy

$$\frac{\sum_{k=1}^R x_k \theta_k}{\sum_{k=1}^R \theta_k} = x_j. \quad (5.84)$$

Zatem

$$\sum_{k=1}^R x_k \theta_k = x_j \sum_{k=1}^R \theta_k \quad (5.85)$$

oraz

$$x_j \theta_j + \sum_{\substack{k=1 \\ k \neq j}}^R x_k \theta_k = x_j \theta_j + x_j \sum_{\substack{k=1 \\ k \neq j}}^R \theta_k. \quad (5.86)$$

W konsekwencji

$$\frac{\sum_{k \neq j} x_k \theta_k}{\sum_{k \neq j} \theta_k} = x_j. \quad (5.87)$$

Stwierdzamy, że warunek konieczny istnienia ekstremum s nie zależy w żaden sposób od parametru θ_k , względem którego liczona była pochodna. Jednakże nierówności (5.82) i (5.83) wskazują, w którym kierunku należy podążać, aby zwiększyć lub zmniejszyć wartość wyrażenia $s(\theta_1, \dots, \theta_R)$. Na podstawie tych nierówności wnioskujemy, że

i) Jeżeli $x_j > s(\theta_1, \dots, \theta_R)$, to $s(\theta_1, \dots, \theta_R)$ rośnie wraz ze zmniejszaniem się parametru θ_j .

ii) Jeżeli $x_j < s(\theta_1, \dots, \theta_R)$, to $s(\theta_1, \dots, \theta_R)$ rośnie wraz ze wzrostem parametru θ_j .

Przypomnijmy, że $\underline{\theta}_k \leq \theta_k \leq \bar{\theta}_k$. Zatem funkcja s osiąga maksimum, jeżeli

a) $\theta_k = \bar{\theta}_k$ dla tych wartości k , dla których $x_k > s$,

b) $\theta_k = \underline{\theta}_k$ dla tych wartości k , dla których $x_k < s$.

Na tej podstawie przedstawimy iteracyjny algorytm poszukiwania maksimum funkcji s :

1) Wyznacz $\theta_k = \frac{\underline{\theta}_k + \bar{\theta}_k}{2}$, $k = 1, \dots, R$, oblicz $s' = s(\theta_1, \dots, \theta_R)$.

2) Znajdź j ($1 \leq j \leq R - 1$) takie, że $x_j \leq s' < x_{j+1}$.

3) Podstaw $\theta_k = \underline{\theta}_k$ dla $k \leq j$ oraz $\theta_k = \bar{\theta}_k$ dla $k > j$.

Oblicz $s'' = s(\underline{\theta}_1, \dots, \underline{\theta}_j, \bar{\theta}_{j+1}, \dots, \bar{\theta}_R)$.

4) Jeżeli $s'' = s'$, to s'' jest maksymalną wartością funkcji s .

Jeżeli $s'' \neq s'$, to przejdź do kroku 5.

5) Podstaw $s' = s''$ i przejdź do kroku 2.

W sposób analogiczny możemy wyznaczyć minimum funkcji s . Funkcja ta osiąga minimum, jeżeli

a) $\theta_k = \bar{\theta}_k$ dla tych wartości k , dla których $x_k < s$,

b) $\theta_k = \underline{\theta}_k$ dla tych wartości k , dla których $x_k > s$.

Iteracyjny algorytm poszukiwania minimum funkcji s przedstawia się następująco:

1) Wyznacz $\theta_k = \frac{\underline{\theta}_k + \bar{\theta}_k}{2}$, $k = 1, \dots, R$, oblicz $s' = s(\theta_1, \dots, \theta_R)$.

2) Znajdź j ($1 \leq j \leq R - 1$) takie, że $x_j < s' \leq x_{j+1}$.

3) Podstaw $\theta_k = \bar{\theta}_k$ dla $k < j$ oraz $\theta_k = \underline{\theta}_k$ dla $k \geq j$, oblicz $s'' = s(\bar{\theta}_1, \dots, \bar{\theta}_j, \underline{\theta}_{j+1}, \dots, \underline{\theta}_R)$.

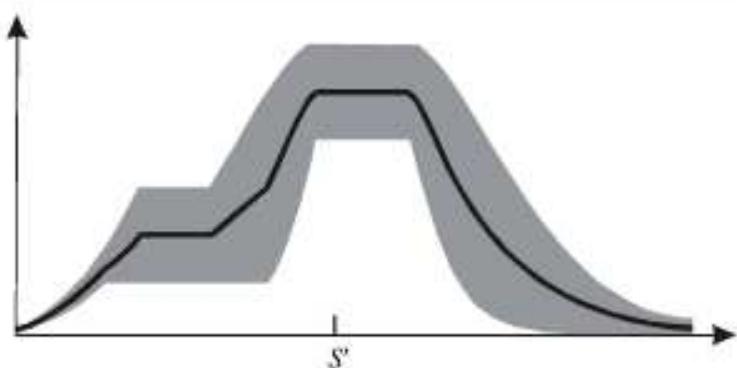
4) Jeżeli $s'' = s'$, to s'' jest minimalną wartością funkcji s .

Jeżeli $s'' \neq s'$, to przejdź do kroku 5.

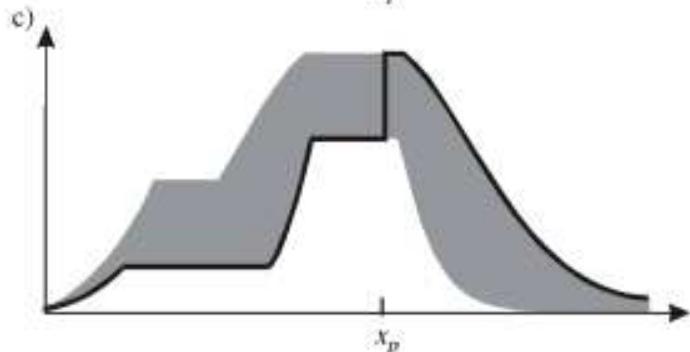
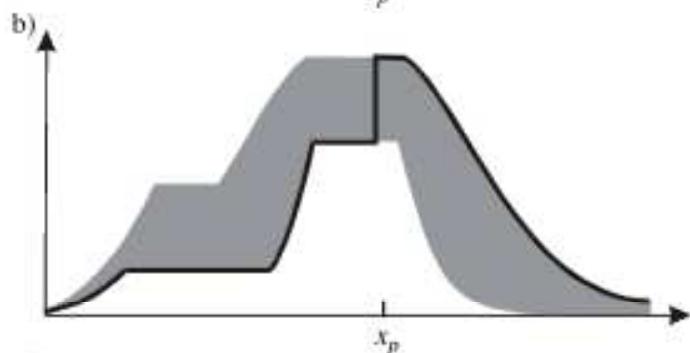
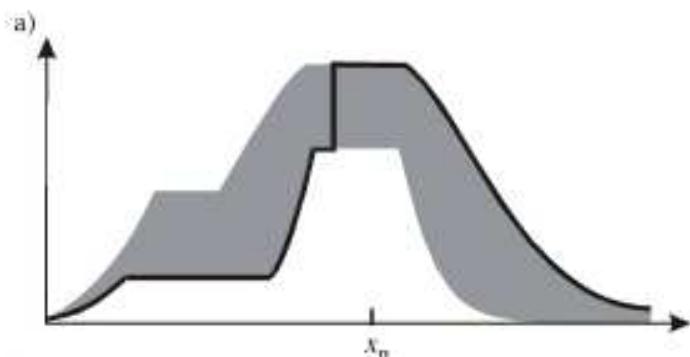
5) Podstaw $s' = s''$ i przejdź do kroku 2.

Przykład 5.16

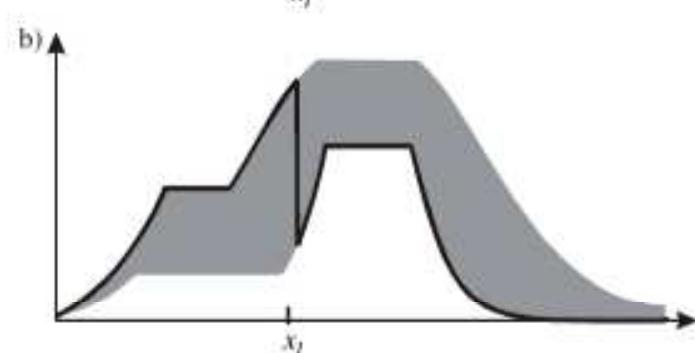
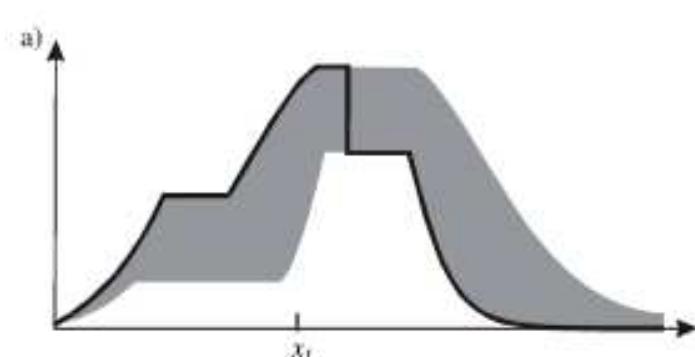
Na rysunkach 5.8, 5.9 i 5.10 przedstawiono sposób działania iteracyjnego algorytmu poszukiwania centroidu zbioru rozmytego typu 2 z przedziałową funkcją drugorzędnej



Rys. 5.8. Zbiór rozmyty typu 2 podlegający wyostrzeniu



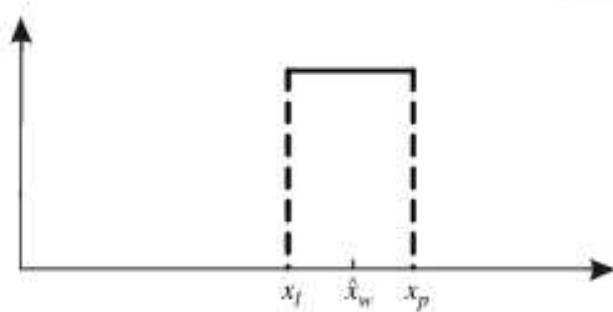
Rys. 5.9. Iteracyjne poszukiwanie punktu x_p



Rys. 5.10. Iteracyjne poszukiwanie punktu x_l

przynależności. Na rysunku 5.8 zaznaczono ślad niepewności zbioru rozmytego typu 2, który będzie podlegał redukcji typu. Linia pogrubiona na tym rysunku koresponduje z punktem 1 iteracyjnego algorytmu, który rozpoczyna się od wyznaczenia środków poszczególnych przedziałów J_x , $x \in X$ oraz wartości wyrażenia (5.77).

Centroid jest zbiorem rozmytym typu 1 danym wzorem (5.78). Iteracyjnie poszukujemy punktu x_p , wyznaczając centroid zbioru rozmytego osadzonego (rys. 5.9), składającego się najpierw z kawałka dolnej funkcji przynależności, a następnie kawałka górnej funkcji przynależności. Podobnie poszukujemy punktu x_l , wyznaczając centroid zbioru rozmytego osadzonego (rys. 5.10), składającego się najpierw z kawałka górnej

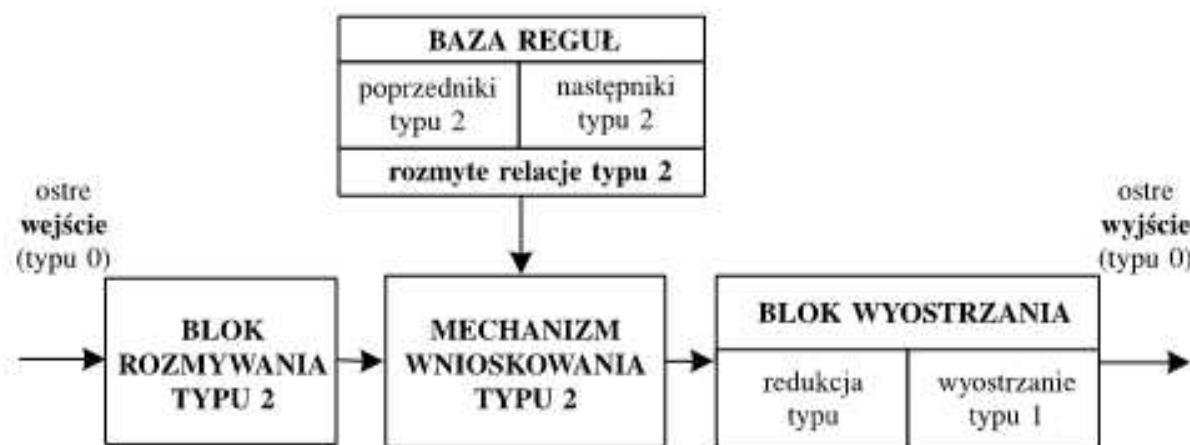
Rys. 5.11. Zbiór rozmyty $C_{\tilde{A}}$

funkcji przynależności, a następnie kawałka dolnej funkcji przynależności. Uzyskany zbiór rozmyty $C_{\tilde{A}} = [x_l, x_p]$ można wyostrzyć (rys. 5.11) w sposób następujący:

$$\hat{x}_w = \frac{x_l + x_p}{2}. \quad (5.88)$$

5.8. Rozmyte systemy wnioskujące typu 2

Rozważmy system rozmyty typu 2 o n zmiennych wejściowych $x_i \in X_i \subset \mathbb{R}, i = 1, \dots, n$ oraz skalarnym wyjściu $y \in \mathbb{Y}$. Na rysunku 5.12 przedstawiono schemat blokowy takiego systemu. Składa się on z następujących elementów: bloku rozmywania typu 2, bazy reguł opisanych relacjami rozmytymi typu 2, mechanizmu wnioskowania typu 2 oraz bloku wyostrzania informacji wyjściowej. Wyostrzanie przebiega tu dwuetapowo:



Rys. 5.12. Rozmyty system wnioskujący typu 2

najpierw następuje redukcja typu, która sprowadza wynik wnioskowania danej reguły typu 2 do zbioru rozmytego typu 1, po czym następuje klasyczne wyostrzanie do zbioru nierożmytego.

5.8.1. Blok rozmywania

Niech $\bar{\mathbf{x}} = (x_1, \dots, x_n)^T \in \mathbf{X} = X_1 \times X_2 \times \dots \times X_n$ będzie sygnałem wejściowym rozmytego systemu wnioskującego. W systemach rozmytych typu 1 stosujemy operację rozmywania typu singleton. Jej odpowiednikiem w systemach rozmytych typu 2 jest

operacja rozmywania typu singleton-singleton zdefiniowana następująco:

$$\tilde{\mu}_{A'}(\mathbf{x}) = \begin{cases} 1/1, & \text{jeżeli } \mathbf{x} = \bar{\mathbf{x}}, \\ 1/0, & \text{jeżeli } \mathbf{x} \neq \bar{\mathbf{x}}. \end{cases} \quad (5.89)$$

W przypadku niezależności poszczególnych zmiennych wejściowych, powyższa operacja przybiera postać

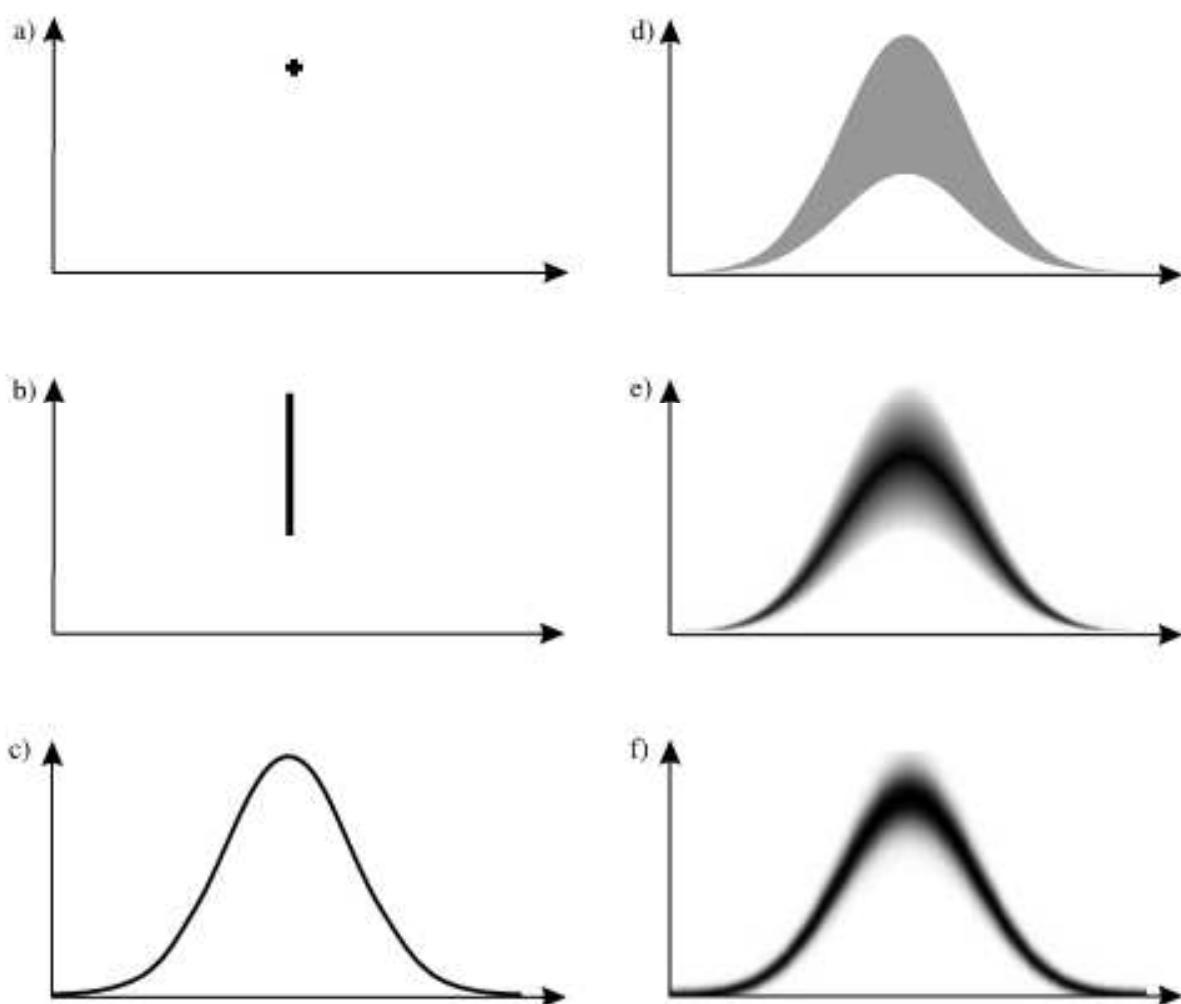
$$\tilde{\mu}_{A'}(x_i) = \begin{cases} 1/1, & \text{jeżeli } x_i = \bar{x}_i, \\ 1/0, & \text{jeżeli } x_i \neq \bar{x}_i \end{cases} \quad (5.90)$$

dla $i = 1, \dots, n$.

W wyniku rozmywania tego typu otrzymujemy n wejściowych zbiorów rozmytych typu 2 postaci

$$\tilde{A}'_i = (1/1)/\bar{x}_i, \quad i = 1, \dots, n, \quad (5.91)$$

gdzie \bar{x}_i jest konkretną wartością i -tej zmiennej wejściowej.



Rys. 5.13. Przykładowe typy rozmywania: a) singleton-singleton, b) singleton-przedział, c) nonsingleton-singleton, d) nonsingleton-przedział, e) nonsingleton-trójkąt, f) nonsingleton-gaussoida

Warto wspomnieć, że możliwe są także inne metody rozmywania sygnału wejściowego. Na rysunku 5.13 przedstawiono ilustrację graficzną tych metod. Na przykład rozmywanie singleton-przedział (rys. 5.13b) oznacza, że drugorzędna funkcja przynależności jest przedziałowym zbiorem rozmytym. Rozmywanie nonsingleton-singleton

(rys. 5.13c) oznacza, że drugorzędna funkcja przynależności jest zbiorem rozmytym typu singleton i w tym przypadku rozmywanie takie jest tożsame z rozmywaniem typu non-singleton dla zbiorów rozmytych typu 1. Rozmywanie nonsingleton-trójkąt (rys. 5.13e) oznacza, że drugorzędna funkcja przynależności jest trójkątnym zbiorem rozmytym, przy czym poziom zacienienia na rysunku 5.13e odzwierciedla wartość drugorzędnej funkcji przynależności (trójkątnej) dla danego elementu $u \in J_x$.

5.8.2. Baza reguł

Model lingwistyczny składa się z N reguł postaci:

$$\tilde{R}^k: \text{JEŻELI } x_1 \text{ jest } \tilde{A}_1^k \text{ I } x_2 \text{ jest } \tilde{A}_2^k \text{ I } \dots \text{ I } x_n \text{ jest } \tilde{A}_n^k \text{ TO } y \text{ jest } \tilde{B}^k, \quad k = 1, \dots, N. \quad (5.92)$$

Oznaczmy

$$\tilde{A} = \tilde{A}^k = \tilde{A}_1^k \times \tilde{A}_2^k \times \dots \times \tilde{A}_n^k. \quad (5.93)$$

Oczywiście

$$\mu_{\tilde{A}^k}(\mathbf{x}) = \prod_{i=1}^n \mu_{\tilde{A}_i^k}(\bar{x}_i). \quad (5.94)$$

Regułę (5.92) można zapisać w postaci implikacji

$$\tilde{A}^k \rightarrow \tilde{B}^k, \quad k = 1, \dots, n. \quad (5.95)$$

5.8.3. Blok wnioskowania

Najpierw wyznaczmy funkcję przynależności $\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y)$. Każda k -ta reguła jest reprezentowana w systemie rozmytym za pomocą pewnej rozmytej relacji typu 2

$$\tilde{R}^k(\mathbf{x}, y) = \int_{\mathbf{X} \times \mathbf{Y}} \mu_{\tilde{R}^k}(\mathbf{x}, y) / (\mathbf{x}, y), \quad (5.96)$$

przy czym

$$\mu_{\tilde{R}^k}(\mathbf{x}, y) = \int_{v \in V_{\mathbf{x}, y}} r_{\mathbf{x}, y}^k(v) / v. \quad (5.97)$$

Zatem

$$\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y) = \mu_{\tilde{R}^k}(\mathbf{x}, y). \quad (5.98)$$

Funkcje przynależności $\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y)$ wyznaczamy, analogicznie jak w przypadku systemów typu 1, na podstawie znajomości funkcji przynależności $\mu_{\tilde{A}^k}(\mathbf{x})$ oraz $\mu_{\tilde{B}^k}(y)$. Za pomocą operatora rozszerzonej t -normy możemy zapisać

$$\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y) = \mu_{\tilde{A}^k}(\mathbf{x}) \bar{*} \mu_{\tilde{B}^k}(y). \quad (5.99)$$

Reguły Mamdaniego oraz Larsena, stosowane w systemach typu 1, obecnie przybierają postać

- rozszerzona reguła Mamdaniego typu minimum

$$\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y) = \int_{u \in J_x^u} \int_{v \in J_y^v} (f_{\mathbf{x}}(u) \stackrel{T}{*} g_y(v)) / \min(u, v), \quad (5.100)$$

- rozszerzona reguła typu iloczyn (Larsena)

$$\mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y) = \int_{u \in J_x^u} \int_{v \in J_y^v} (f_{\mathbf{x}}(u) \stackrel{T}{*} g_y(v)) / uv. \quad (5.101)$$

Na wyjściu bloku wnioskowania otrzymujemy zbiór rozmyty typu 2 \tilde{B}^k . Zbiór ten jest określony przez złożenie wejściowego zbioru rozmytego \tilde{A}^k i rozmytej relacji \tilde{R}^k , tzn.

$$\tilde{B}^k = \tilde{A}^k \circ \tilde{R}^k = \tilde{A}^k \circ (\tilde{A}^k \rightarrow \tilde{B}^k). \quad (5.102)$$

Korzystając z definicji 5.11, wyznaczamy funkcję przynależności zbioru rozmytego \tilde{B}^k

$$\begin{aligned} \mu_{\tilde{B}^k}(y) &= \mu_{\tilde{A}^k \circ \tilde{R}^k}(y) = \tilde{S}_{\mathbf{x} \in \mathbf{X}} (\mu_{\tilde{A}^k}(\mathbf{x}) \stackrel{T}{*} \mu_{\tilde{R}^k}(\mathbf{x}, y)) \\ &= \tilde{S}_{\mathbf{x} \in \mathbf{X}} (\mu_{\tilde{A}^k}(\mathbf{x}) \stackrel{T}{*} \mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\mathbf{x}, y)). \end{aligned} \quad (5.103)$$

W przypadku rozmywania typu singleton-singleton (5.84) powyższy wzór przybiera postać

$$\mu_{\tilde{B}^k}(y) = \mu_{\tilde{A}^k \rightarrow \tilde{B}^k}(\bar{\mathbf{x}}, y). \quad (5.104)$$

Korzystając ze wzorów (5.99) oraz (5.94), otrzymujemy

$$\mu_{\tilde{B}^k}(y) = \mu_{\tilde{A}_1^k \times \dots \times \tilde{A}_n^k}(\bar{\mathbf{x}}) \stackrel{T}{*} \mu_{\tilde{B}^k}(y) = \left(\tilde{T}_{i=1}^n \mu_{\tilde{A}_i^k}(\bar{x}_i) \right) \stackrel{T}{*} \mu_{\tilde{B}^k}(y). \quad (5.105)$$

Oznaczmy stopień aktywacji k -tej reguły w sposób następujący:

$$\tau_k = \tilde{T}_{i=1}^n \mu_{\tilde{A}_i^k}(\bar{x}_i). \quad (5.106)$$

Wówczas zależność (5.105) przybierze postać

$$\mu_{\tilde{B}^k}(y) = \tau_k \stackrel{T}{*} \mu_{\tilde{B}^k}(y). \quad (5.107)$$

Uwaga 5.7

W przypadku zbiorów rozmytych typu 1 stopień aktywacji reguły τ_k jest liczbą rzeczywistą, przy czym $\tau_k \in [0, 1]$. W przypadku zbiorów rozmytych typu 2 stopień aktywacji reguły τ_k jest zbiorem rozmytym typu 1.

Mając rezultaty wnioskowania \tilde{B}^k dla wszystkich N reguł, dokonujemy agregacji, korzystając z operatora rozszerzonej t -konormy

$$\mu_{\tilde{B}^k}(y) = \tilde{S}_{k=1}^N \mu_{\tilde{B}^k}(y). \quad (5.108)$$

Pokażemy teraz, jak przebiega proces wnioskowania w systemach przedziałowych. W takich systemach funkcje drugorzędnej przynależności zbiorów rozmytych \tilde{A}_i^k oraz \tilde{B}^k ,

$i = 1, \dots, n, k = 1, \dots, N$, są funkcjami stałymi, przyjmującymi wartość 1 we wszystkich przedziałach $J_x, x \in X$. W dalszych rozważaniach wykorzystamy dwie właściwości ([97, 134]) interwałowych zbiorów rozmytych typu 1, F_1, \dots, F_n , zdefiniowanych na przedziałach $[l_1, p_1], \dots, [l_n, p_n]$, przy czym $l_i \geq 0$ oraz $p_i \geq 0, i = 1, \dots, n$.

1) Rozszerzona t -norma $\tilde{T}_{i=1}^n F_i$ jest interwałowym zbiorem rozmytym typu 1 zdefiniowanym na przedziale $[(l_1 * l_2 * \dots * l_n), (p_1 * p_2 * \dots * p_n)]$, gdzie $*$ oznacza t -normę typu minimum lub iloczyn.

2) Rozszerzona t -konorma $\tilde{S}_{i=1}^n F_i$ jest interwałowym zbiorem rozmytym typu 1 zdefiniowanym na przedziale $[(l_1 \vee l_2 \vee \dots \vee l_n), (p_1 \vee p_2 \vee \dots \vee p_n)]$, gdzie \vee oznacza operację maksimum.

Wprowadzimy symboliczną notację, zgodnie z którą przedziałowy zbiór rozmyty A zapisujemy jako

$$A = \int_{x \in [a, b]} 1/x \equiv [a, b]. \quad (5.109)$$

Korzystając z właściwości 1, wyrazimy stopień aktywacji reguły τ_k , będący teraz przedziałowym zbiorem rozmytym typu 1, poprzez wartości dolnych i górnych funkcji przynależności zbiorów rozmytych \tilde{A}_i^k . Na podstawie właściwości 1 możemy zapisać

$$\tau_k = [\underline{\tau}_k, \bar{\tau}_k], \quad (5.110)$$

gdzie

$$\underline{\tau}_k(\bar{x}) = \underline{\mu}_{\tilde{A}_1^k}(\bar{x}_1) * \dots * \underline{\mu}_{\tilde{A}_n^k}(\bar{x}_n) \quad (5.111)$$

oraz

$$\bar{\tau}_k(\bar{x}) = \bar{\mu}_{\tilde{A}_1^k}(\bar{x}_1) * \dots * \bar{\mu}_{\tilde{A}_n^k}(\bar{x}_n). \quad (5.112)$$

Korzystając ze wzorów (5.107), (5.110) oraz właściwości 1, otrzymujemy

$$\mu_{\tilde{B}^k}(y) = \mu_{\tilde{B}^k}(y) * [\underline{\tau}^k, \bar{\tau}^k] \equiv [\underline{b}^k(y), \bar{b}^k(y)], \quad y \in Y, \quad (5.113)$$

gdzie

$$\underline{b}^k(y) = \underline{\tau}^k * \underline{\mu}_{\tilde{B}^k}(y) \quad (5.114)$$

oraz

$$\bar{b}^k(y) = \bar{\tau}^k * \bar{\mu}_{\tilde{B}^k}(y). \quad (5.115)$$

Korzystając ze wzorów (5.113) oraz (5.108) i właściwości 2, możemy wyznaczyć

$$\mu_{\tilde{B}^k}(y) = \sum_{k=1}^N \mu_{\tilde{B}^k}(y) = \sum_{k=1}^N [\underline{b}^k(y), \bar{b}^k(y)] = [\underline{b}(y), \bar{b}(y)], \quad (5.116)$$

gdzie

$$\underline{b}(y) = \underline{b}^1(y) \vee \underline{b}^2(y) \vee \dots \vee \underline{b}^N(y) \quad (5.117)$$

oraz

$$\bar{b}(y) = \bar{b}^1(y) \vee \bar{b}^2(y) \vee \dots \vee \bar{b}^N(y). \quad (5.118)$$

Przykład 5.17

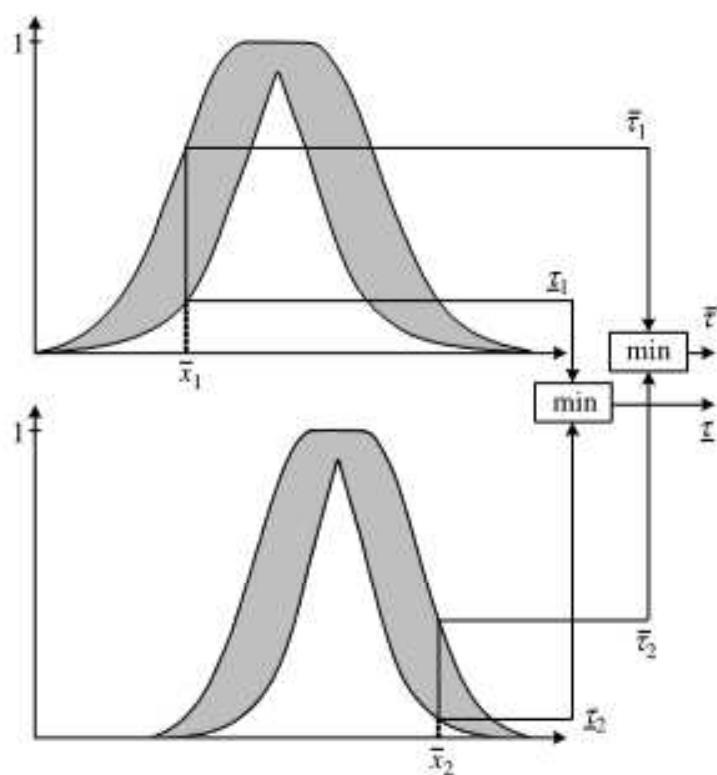
Na rysunku 5.14 pokazano sposób wyznaczania stopni aktywacji systemu typu 2 z dwiema regułami. Jako t -normę przyjęto operacje minimum, a zatem

$$\underline{\tau}_k = \min \left[\mu_{\tilde{A}_k^1}(\bar{x}_1), \mu_{\tilde{A}_k^2}(\bar{x}_2) \right] \quad (5.119)$$

oraz

$$\bar{\tau}_k = \min \left[\bar{\mu}_{\tilde{A}_k^1}(\bar{x}_1), \bar{\mu}_{\tilde{A}_k^2}(\bar{x}_2) \right] \quad (5.120)$$

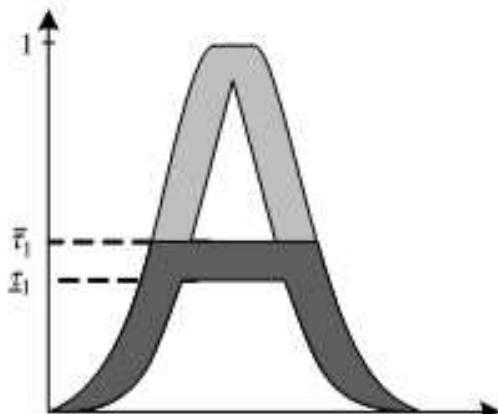
dla $k = 1, 2$. Jak wcześniej podkreślaliśmy, stopnie aktywacji są przedziałowymi zbiorami rozmytymi typu 1.



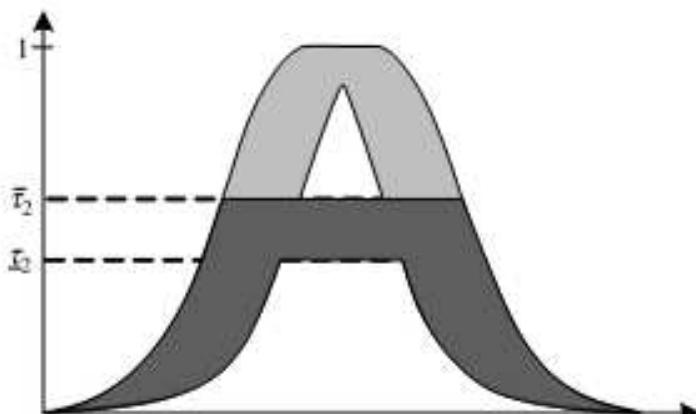
Rys. 5.14. Sposób wyznaczenia stopni aktywacji przedziałowego systemu rozmytego typu 2 z rozmywaniem singleton-singleton

Przykład 5.18

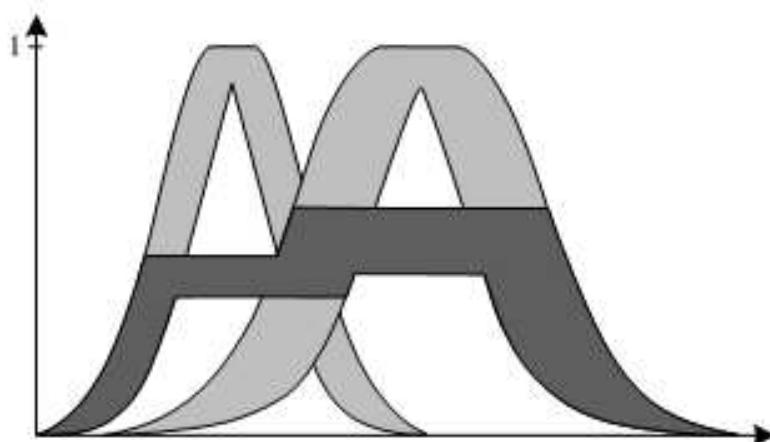
Na rysunkach 5.15 i 5.16 pokazano wyjściowe zbiory rozmyte typu 2 \widetilde{B}^1 i \widetilde{B}^2 , jak również zbiory rozmyte (zacielenione) $\widetilde{B}^{1\prime}$ i $\widetilde{B}^{2\prime}$, będące wynikiem wnioskowania opisanego wzorem



Rys. 5.15. Zbiory rozmyte typu 2 \tilde{B}^1 i \tilde{B}^{11}



Rys. 5.16. Zbiory rozmyte typu 2 \tilde{B}^2 i \tilde{B}^2

Rys. 5.17. Zbiór rozmyty typu 2 \tilde{B}' (agregacja zbiorów \tilde{B}'^1 i \tilde{B}'^2)

(5.113). Na rysunku 5.17 przedstawiono zbiór rozmyty (zaciemiony) \tilde{B}' , określony wzorem (5.116) i będący wynikiem agregacji zbiorów rozmytych \tilde{B}'^1 i \tilde{B}'^2 . W celu wyznaczenia tego zbioru zastosowaliśmy operacje

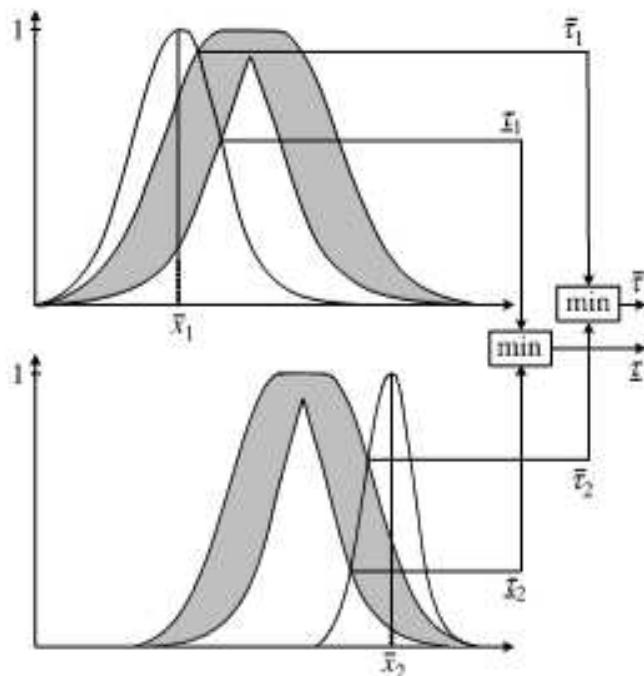
$$\max \left(\min \bar{\tau}_1, \bar{\mu}_{\tilde{B}^1}(y), \min \bar{\tau}_2, \bar{\mu}_{\tilde{B}^2}(y) \right) \quad (5.121)$$

oraz

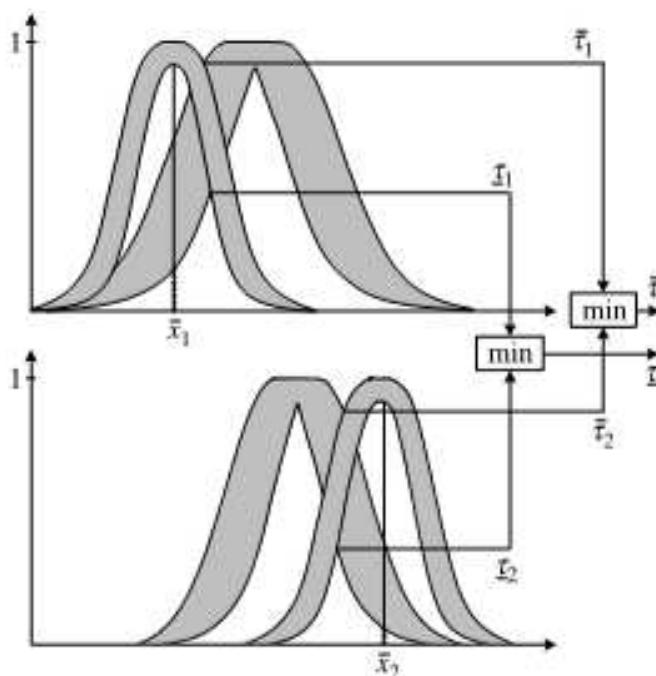
$$\max \left(\min \underline{\tau}_1, \underline{\mu}_{\tilde{B}^1}(y), \min \underline{\tau}_2, \underline{\mu}_{\tilde{B}^2}(y) \right). \quad (5.122)$$

Przykład 5.19

W przykładach 5.17 i 5.18 przedstawiono rezultaty uzyskane dla przedziałowych systemów rozmytych typu 2 z rozmywaniem typu singleton danym wzorem (5.89). Rezultaty te można uogólnić na przypadek, gdy sygnałem wejściowym jest zbiór rozmyty typu 1



Rys. 5.18. Sposób wyznaczenia stopni aktywacji przedziałowego systemu rozmytego typu 2 z rozmywaniem nonsingleton-singleton



Rys. 5.19. Sposób wyznaczenia stopni aktywacji przedziałowego systemu rozmytego typu 2 z rozmywaniem nonsingleton typu 2-przedział

(rozmywanie nonsingleton-singleton) lub przedziałowy zbiór rozmyty typu 2 (rozmywanie nonsingleton typu 2–przedział). Na rysunkach 5.18 i 5.19 przedstawiono sposób wyznaczania stopni aktywacji ilustrujący oba przypadki.

5.9. Uwagi

Pojęcie zbioru rozmytego typu 2 wprowadził Lotfi Zadeh [266]. W artykule tym autor definiuje też sumę i przecięcie zbiorów rozmytych typu 2, wykorzystując do tego celu zasadę rozszerzania. Podstawowe pojęcia charakteryzujące zbiory rozmyte typu 2, tj. funkcje i stopnie drugorzędnej przynależności, funkcje głównej, górnej i dolnej przynależności, jak również pojęcia osadzonych zbiorów rozmytych oraz śladu niepewności, sukcesywnie wprowadzał do literatury światowej Mendel, a ich przegląd zawarty jest w monografii [134]. Metodę wnioskowania z użyciem przedziałowych zbiorów rozmytych typu 2 jako pierwszy opisał Gorzałczany [64]. Podstawowe operacje na zbiorach rozmytych typu 2 podali Dubois i Prade [42] oraz Karnik i Mendel [97, 100]. Przedziałowe zbiory rozmyte wyższych rzędów badał Hisdal [80]. Iteracyjny algorytm redukcji typu dla przedziałowych zbiorów rozmytych typu 2 wprowadzili Karnik i Mendel [97, 101]. Pozwoliło to na skonstruowanie przedziałowych systemów rozmytej logiki typu 2. Pierwsze takie konstrukcje zaprezentowali Karnik, Mendel i Liang [99]. Analizie różnic pomiędzy przedziałowymi systemami wnioskującymi a systemami typu 1 poświęcony został artykuł Starczewskiego [240]. Interesującą metodę redukcji typu podali Wu i Mendel w artykule [261]. Przedziałowe systemy typu 2 zastosowano do predykcji szeregów chaotycznych [98]. Nowością jest konstrukcja systemu rozmytego wnioskowania typu 2 z trójkątną funkcją drugorzędnej przynależności przedstawiona przez Starczewskiego [238].

Sieci neuronowe i algorytmy ich uczenia

6.1. Wprowadzenie

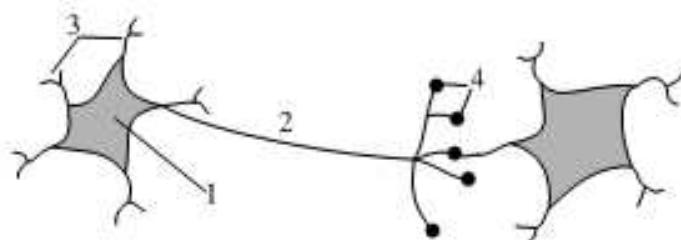
Naukowcy od wielu lat próbują poznać budowę i sposób działania mózgu. Niestety, do dziś jest on nie do końca rozwiązana, fascynującą zagadką. Na podstawie obserwacji ludzi okaleczonych w różnych wojnach czy wypadkach naukowcy mogli ocenić specjalizację poszczególnych fragmentów mózgu. Stwierdzono na przykład, że lewa półkula jest odpowiedzialna za sterowanie prawą ręką, natomiast prawa półkula — lewą ręką. Naukowcy nie mają jeszcze dokładnych informacji na temat wyższych czynności psychicznych. Można przyjąć hipotetycznie, iż lewa półkula kontroluje funkcje mowy i ścisłe myślenie, prawa półkula natomiast jest jej przeciwnieństwem, gdyż w jej gestii są zdolności artystyczne, wyobraźnia przestrzenna itp. System nerwowy zbudowany jest z komórek nazywanych neuronami. Ich liczba w ludzkim mózgu wynosi ok. 100 miliardów. Działanie pojedynczego neuronu polega na przepływie tzw. impulsów nerwowych. Impuls wytworzony przez określony bodziec, trafiając na neuron, powoduje rozprzestrzenienie się go wzdłuż wszystkich jego wypustek. W wyniku tego może nastąpić skurcz mięśnia lub pobudzenie kolejnego neuronu. Czemu więc odpowiednio połączone sztuczne neurony nie mogłyby, zamiast sterować mięśniami, kierować na przykład pracą jakiegoś urządzenia lub rozwiązywać różne problemy wymagające inteligencji? Ten rozdział poświęcony jest sztucznym sieciom neuronowym. Przedstawimy modele matematyczne pojedynczego neuronu, rozmaite struktury sztucznych sieci neuronowych oraz algorytmy ich uczenia.

6.2. Neuron i jego modele

6.2.1. Budowa i działanie pojedynczego neuronu

Podstawowym elementem systemu nerwowego jest komórka nerwowa nazywana *neuronem*. Na rysunku 6.1 pokazany jest jej uproszczony schemat. W neuronie możemy wyróżnić *ciało komórki* (nazywane somą) oraz otaczające je dwa rodzaje wypustek: wypustki wprowadzające informację do neuronu, tzw. *dendryty* i wypustkę wyprowadzającą informacje z neuronu, tzw. *akson*. Każdy neuron ma dokładnie jedną wypustkę

wyprowadzającą, poprzez którą może wysyłać impulsy do wielu innych neuronów. Pojedynczy neuron przyjmuje pobudzenie od ogromnej liczby neuronów dochodzącej do tysiąca. Jak wspomnieliśmy wcześniej, w mózgu człowieka jest ok. 100 miliardów neuronów, które oddziałują na siebie poprzez ogromną liczbę połączeń. Jeden neuron przekazuje pobudzenie innym neuronom przez złącza nerwowe nazywane *synapsami*, przy czym transmisja sygnałów odbywa się na drodze skomplikowanych procesów chemiczno-elektrycznych. Synapsy pełnią funkcję przekaźników informacji, a w wyniku ich działania pobudzenie może być wzmacnione lub osłabione. W rezultacie do neuronu dochodzą sygnały, z których część wywiera wpływ pobudzający, a część hamujący. Neuron sumuje impulsy pobudzające i hamujące. Jeżeli ich suma algebraiczna przekracza pewną wartość progową, to sygnał na wyjściu neuronu jest przesyłany — poprzez akson — do innych neuronów.



Rys. 6.1. Uproszczony schemat neuronu i jego połączenia z sąsiednim neuronem: 1 — ciało komórki, 2 — akson, 3 — dendryty, 4 — synapsy

Przedstawimy teraz model neuronu nawiązujący do pierwszych prób sformalizowania opisu działania komórki nerwowej. Wprowadźmy następujące oznaczenia: n — liczba wejść w neuronie, x_1, \dots, x_n — sygnały wejściowe, $\mathbf{x} = [x_1, \dots, x_n]^T$, w_0, \dots, w_n — wag synaptyczne, $\mathbf{w} = [w_0, \dots, w_n]^T$, y — wartość wyjściowa neuronu, w_0 — wartość progowa, f — funkcja aktywacji.

Formuła opisująca działanie neuronu wyraża się zależnością

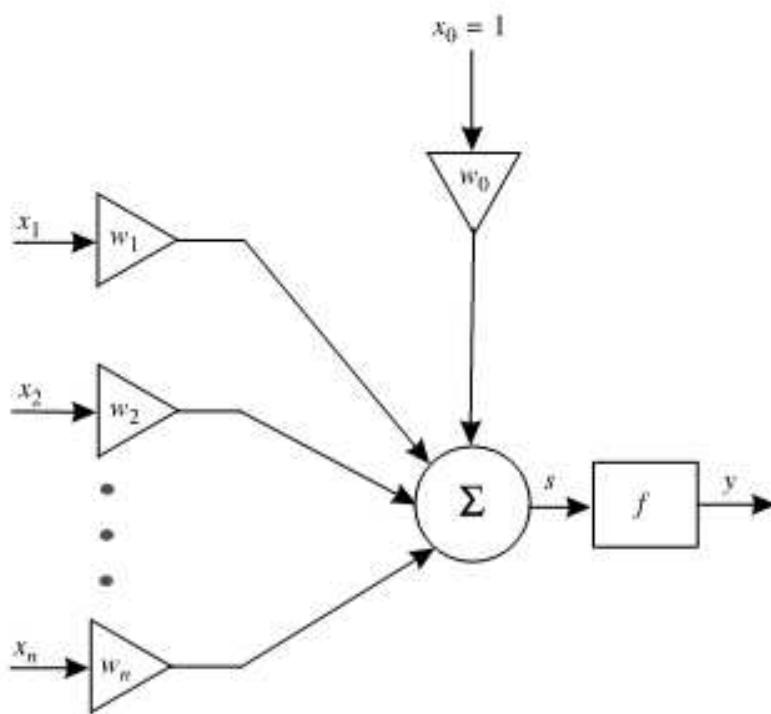
$$y = f(s), \quad (6.1)$$

w której

$$s = \sum_{i=0}^n x_i w_i. \quad (6.2)$$

Wzory (6.1) i (6.2) opisują neuron przedstawiony na rysunku 6.2. Funkcja aktywacji f może przybierać różną postać w zależności od konkretnego modelu neuronu.

Jak wynika z powyższych wzorów, działanie neuronu jest bardzo proste. Najpierw sygnały wejściowe x_0, x_1, \dots, x_n zostają pomnożone przez odpowiadające im wagę w_0, w_1, \dots, w_n . Otrzymane w ten sposób wartości należy następnie zsumować. W wyniku powstaje sygnał s odzwierciedlający działanie części liniowej neuronu. Sygnał ten jest poddawany działaniu funkcji aktywacji, najczęściej nieliniowej. Zakładamy, że wartość sygnału x_0 jest równa 1, natomiast wagę w_0 nazywa się *progiem* (ang. *bias*). Gdzie zatem kryje się wiedza w tak opisanym neuronie? Otóż wiedza zapisana jest właśnie w wagach. Największym zaś fenomenem jest to, iż w łatwy sposób (za pomocą algorytmów opisanych w dalszej części rozdziału) można neurony uczyć, a więc odpowiednio dobierać wagę. Na rysunku 6.2 przedstawiliśmy ogólny schemat neuronu, jednakże w sieciach

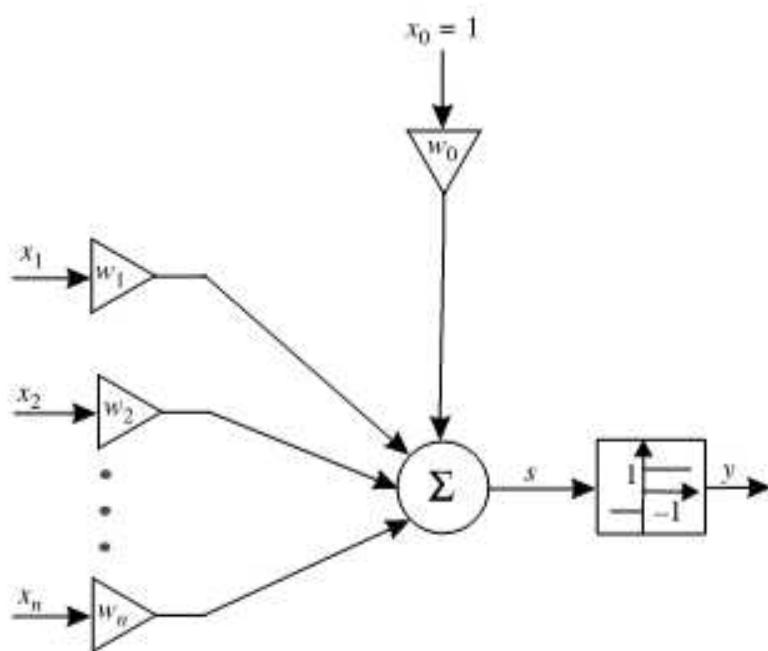


Rys. 6.2. Model neuronu

stosuje się różne jego modele. Niektóre z nich omówimy w następnych punktach. Należy jeszcze wspomnieć, iż podobnie jak w mózgu komórki nerwowe łączą się ze sobą, tak i w przypadku tworzenia modeli matematycznych sztuczne neurony przedstawione na rysunku 6.2 łączy się ze sobą, tworząc wielowarstwowe sieci neuronowe. Sposób łączenia neuronów, a także metody uczenia powstałych w ten sposób struktur poznamy w kolejnych punktach tego rozdziału.

6.2.2. Perceptron

Na rysunku 6.3 przedstawiono schemat perceptronu.



Rys. 6.3. Schemat perceptronu

Działanie perceptronu można opisać zależnością

$$y = f\left(\sum_{i=1}^n w_i x_i + \theta\right). \quad (6.3)$$

Zauważmy, że wzór (6.3) koresponduje z ogólnym zapisem (6.1), jeżeli $\theta = w_0$. Funkcja f może być nieciągłą funkcją skokową — bipolarną (przyjmuje wartości -1 lub 1) lub unipolarną (przyjmuje wartości 0 lub 1). Do dalszych rozważań przyjmiemy, iż funkcja aktywacji jest bipolarna

$$f(s) = \begin{cases} 1, & \text{gdy } s > 0, \\ -1, & \text{gdy } s \leq 0. \end{cases} \quad (6.4)$$

Perceptron ze względu na swoją funkcję aktywacji przyjmuje tylko dwie różne wartości wyjściowe, może więc klasyfikować sygnały podane na jego wejście w postaci wektorów $\mathbf{x} = [x_1, \dots, x_n]^T$ do jednej z dwóch klas. Na przykład perceptron z jednym wejściem może oceniać, czy sygnał wejściowy jest dodatni, czy ujemny. W przypadku dwóch wejść x_1 i x_2 perceptron dzieli płaszczyznę na dwie części. Podział ten wyznacza prostą o równaniu

$$w_1 x_1 + w_2 x_2 + \theta = 0. \quad (6.5)$$

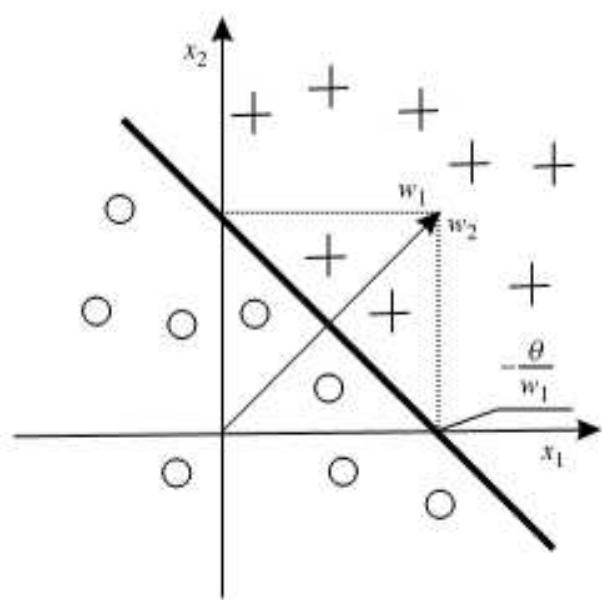
Zatem równanie (6.5) można napisać

$$x_2 = -\frac{w_1}{w_2} \cdot x_1 - \frac{\theta}{w_2}. \quad (6.6)$$

W ogólnym przypadku, gdy perceptron ma n wejść, wówczas dzieli n -wymiarową przestrzeń wektorów wejściowych \mathbf{x} na dwie półprzestrzenie. Są one rozdzielone $n-1$ -wymiarową hiperpłaszczyzną, nazywaną *granicą decyzyjną*, daną wzorem

$$\sum_{i=1}^n w_i x_i + \theta = 0. \quad (6.7)$$

Na rysunku 6.4 przedstawiono granicę decyzyjną dla $n = 2$. Należy zauważyc, że prosta wyznaczająca podział przestrzeni jest zawsze prostopadła do wektora wag $\mathbf{w} = [w_1, w_2]^T$.



Rys. 6.4. Granica decyzyjna dla $n = 2$

Zgodnie z tym, co napisaliśmy we wstępie, perceptron można uczyć. W czasie tego procesu jego wagi są modyfikowane. Metoda uczenia perceptronu należy do grupy algorytmów zwanych *uczeniem z nauczycielem* lub *uczeniem nadzorowanym*. Uczenie tego typu polega na tym, iż podaje się na wejście perceptronu sygnały $\mathbf{x}(t) = [x_0(t), x_1(t), \dots, x_n(t)]^T$, $t = 1, 2, \dots$, dla których znamy prawidłowe wartości sygnałów wyjściowych $d(t)$, $t = 1, 2, \dots$, zwanych sygnałami wzorcowymi. Zbiór takich próbek wejściowych wraz z odpowiadającymi im wartościami sygnałów wzorcowych nazywamy *ciągiem uczącym*. W metodach tych po podaniu wartości wejściowych oblicza się sygnał wyjściowy neuronu. Następnie modyfikuje się wagi w ten sposób, aby minimalizować błąd między sygnałem wzorcowym a wyjściem perceptronu. Stąd właśnie nazwa „uczenie z nauczycielem”, ponieważ nauczyciel określa, jaka powinna być wartość wzorcowa. Oczywiście można się domyślać, iż istnieją algorytmy uczące sieci bez nauczyciela, ale o nich napiszemy w następnych punktach tego rozdziału. Algorytm uczenia perceptronu przedstawia się następująco:

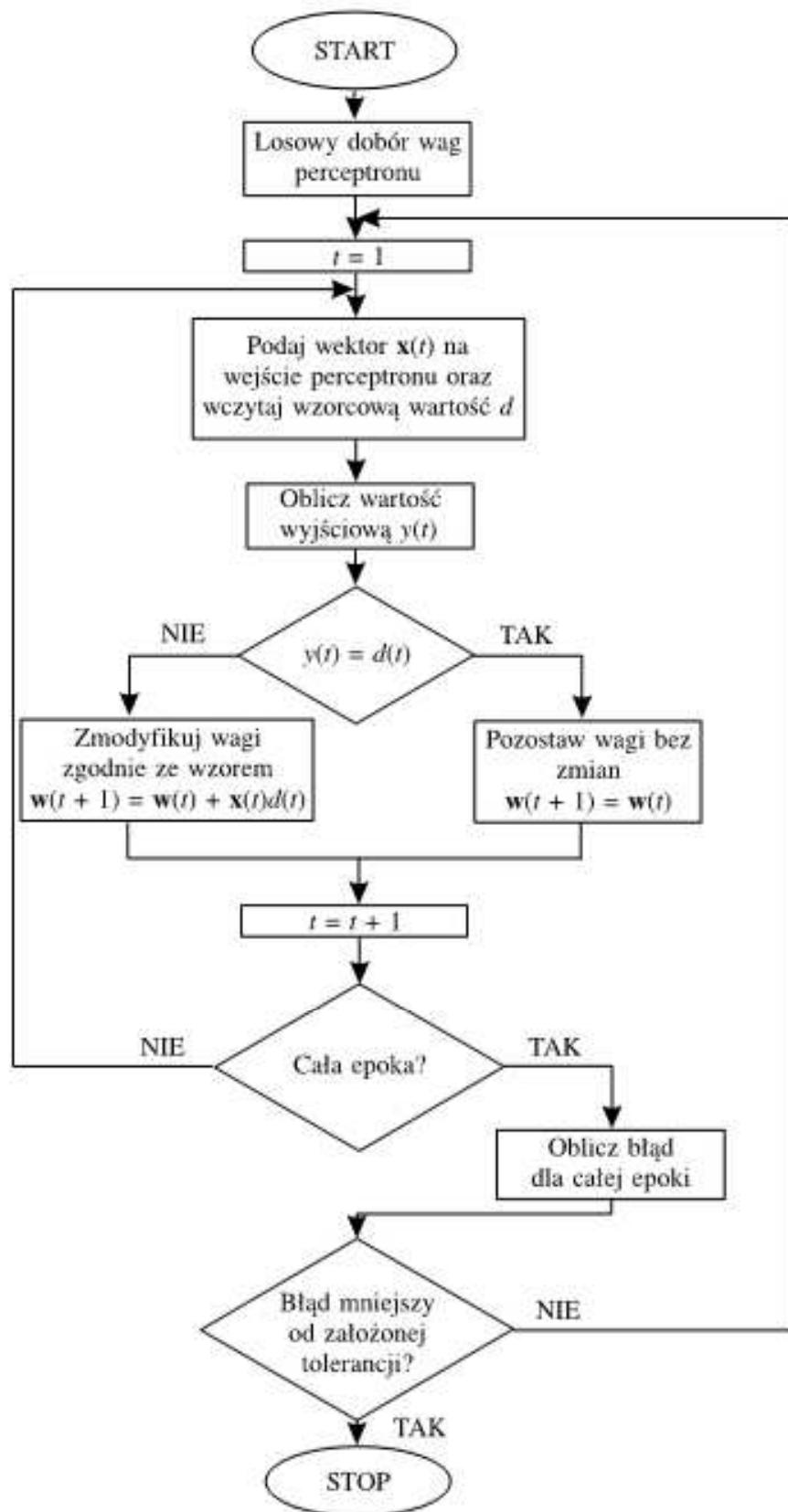
1. Wybieramy w sposób losowy wagi początkowe perceptronu.
2. Na wejścia neuronu podajemy wektor uczący \mathbf{x} , przy czym $\mathbf{x} = \mathbf{x}(t) = [x_0(t), x_1(t), \dots, x_n(t)]^T$, $t = 1, 2, \dots$
3. Obliczamy wartość wyjściową perceptronu y , zgodnie ze wzorem (6.3).
4. Porównujemy wartość wyjściową $y(t)$ z wartością wzorcową $d = d(\mathbf{x}(t))$ znajdującą się w ciągu uczącym.
5. Dokonujemy modyfikacji wag według zależności:
 - a) jeżeli $y(\mathbf{x}(t)) \neq d(\mathbf{x}(t))$, to $w_i(t+1) = w_i(t) + d(\mathbf{x}(t))x_i(t)$;
 - b) jeżeli $y(\mathbf{x}(t)) = d(\mathbf{x}(t))$, to $w_i(t+1) = w_i(t)$, czyli wagi pozostają bez zmian.
6. Wracamy do punktu 2.

Algorytm powtarza się tak długo, aż dla wszystkich wektorów wejściowych wchodzących w skład ciągu uczącego błąd na wyjściu będzie mniejszy od założonej tolerancji. Na rysunku 6.5 przedstawiono schemat blokowy uczenia perceptronu. Działanie pętli wewnętrznej na tym rysunku dotyczy tzw. jednej *epoki*, na którą składają się dane tworzące ciąg uczący. Działanie pętli zewnętrznej odzwierciedla możliwość wielokrotnego stosowania tego samego ciągu uczącego, aż zostanie spełniony warunek zatrzymania algorytmu.

Wykażemy, iż algorytm uczenia perceptronu jest zbieżny. Twierdzenie o zbieżności algorytmu uczenia perceptronu formułuje się następująco:

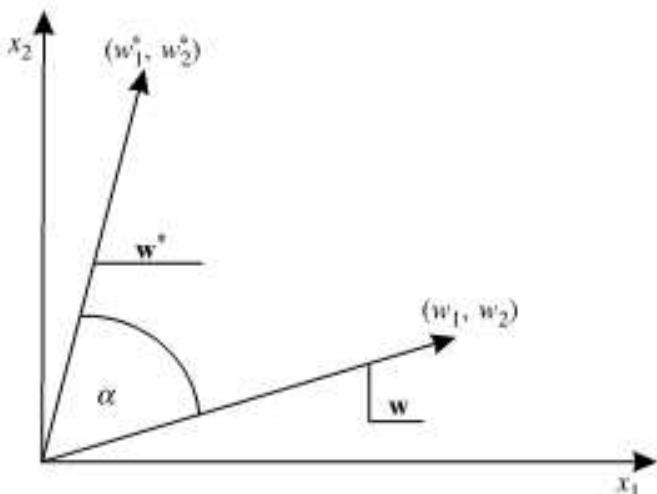
Jeżeli istnieje zestaw wag $\mathbf{w}^* = [w_1^*, \dots, w_n^*]^T$ klasyfikujący wzorce uczące $\mathbf{x} = [x_1, \dots, x_n]^T$ poprawnie, tzn. wyznaczający odwzorowanie $y = d(\mathbf{x})$, to algorytm uczący znajdzie rozwiązanie w skończonej liczbie iteracji dla dowolnych wartości początkowych wektora wag \mathbf{w} .

Zakładamy, że dane uczące reprezentują klasy liniowo separowane, gdyż tylko wtedy można nauczyć perceptron. Pokażemy, że istnieje skończona liczba kroków modyfikacji wag, po których perceptron będzie realizował odwzorowanie $y = d(\mathbf{x})$. Ze względu na to, że funkcja aktywacji w perceptronie jest typu sgn, długość wektora \mathbf{w}^* możemy przyjąć dowolną, np. równą 1, tzn. $\|\mathbf{w}^*\| = 1$. Zatem w czasie uczenia wektor \mathbf{w}



Rys. 6.5. Schemat blokowy algorytmu uczenia perceptronu

wystarczy modyfikować tak, aby pokazany na rysunku 6.6 kąt α był równy 0. Oczywiście wtedy $\cos(\alpha) = 1$. Z faktu, że $|\mathbf{w}^* \circ \mathbf{x}| > 0$ (znak \circ w tym przypadku oznacza iloczyn skalarny wektorów) i \mathbf{w}^* jest rozwiązaniem, wynika istnienie takiej stałej $\delta > 0$, dla której $|\mathbf{w}^* \circ \mathbf{x}| > \delta$ dla wszystkich wektorów \mathbf{x} z ciągu uczącego. Z definicji iloczynu



Rys. 6.6. Ilustracja działania algorytmu uczenia perceptronu dla $n = 2$

skalarnego wynika, że

$$\cos(\alpha) = \frac{\mathbf{w}^* \circ \mathbf{w}}{\sqrt{\|\mathbf{w}^*\|^2 \|\mathbf{w}\|^2}}. \quad (6.8)$$

Ponieważ

$$\sqrt{\|\mathbf{w}^*\|^2} = \|\mathbf{w}^*\| = 1, \quad (6.9)$$

więc

$$\cos(\alpha) = \frac{\mathbf{w}^* \circ \mathbf{w}}{\|\mathbf{w}\|}. \quad (6.10)$$

Zgodnie z algorytmem uczenia perceptronu, wagi modyfikowane są dla podanego wektora wejściowego \mathbf{x} według następującej zależności: $\mathbf{w}' = \mathbf{w} + \Delta\mathbf{w}$, gdzie $\Delta\mathbf{w} = d(\mathbf{x})\mathbf{x}$. Oczywiście zakładamy, że na wyjściu sieci pojawił się błąd i korekcja wag jest niezbędna. Zauważmy, że

$$\mathbf{w}' \circ \mathbf{w}^* = \mathbf{w} \circ \mathbf{w}^* + d(\mathbf{x})\mathbf{w}^* \circ \mathbf{x} \quad (6.11)$$

$$= \mathbf{w} \circ \mathbf{w}^* + \text{sgn}(\mathbf{w} \circ \mathbf{x})\mathbf{w}^* \circ \mathbf{x}. \quad (6.12)$$

Zachodzą następujące fakty:

- (i) Jeżeli $\mathbf{w}^* \circ \mathbf{x} < 0$, to $\text{sgn}(\mathbf{w}^* \circ \mathbf{x}) = -1$, więc $\text{sgn}(\mathbf{w}^* \circ \mathbf{x})\mathbf{w}^* \circ \mathbf{x} = -1(\mathbf{w}^* \circ \mathbf{x}) > 0$,
- (ii) Jeżeli $\mathbf{w}^* \circ \mathbf{x} > 0$, to $\text{sgn}(\mathbf{w}^* \circ \mathbf{x}) = 1$, więc $\text{sgn}(\mathbf{w}^* \circ \mathbf{x})\mathbf{w}^* \circ \mathbf{x} = 1(\mathbf{w}^* \circ \mathbf{x}) > 0$.

Zatem

$$\text{sgn}(\mathbf{w}^* \circ \mathbf{x})\mathbf{w}^* \circ \mathbf{x} = |\mathbf{w}^* \circ \mathbf{x}|. \quad (6.13)$$

Zgodnie ze wzorami (6.12) i (6.13) możemy napisać

$$\mathbf{w}' \circ \mathbf{w}^* = \mathbf{w} \circ \mathbf{w}^* + |\mathbf{w}^* \circ \mathbf{x}|. \quad (6.14)$$

Wiemy też, że $|\mathbf{w}^* \circ \mathbf{x}| > \delta$, stąd

$$\mathbf{w}' \circ \mathbf{w}^* > \mathbf{w} \circ \mathbf{w}^* + \delta. \quad (6.15)$$

Oszacujmy teraz wartość $\|\mathbf{w}'\|^2$, pamiętając jednocześnie, iż rozpatrujemy przypadek, kiedy po podaniu na wejście wektora uczącego \mathbf{x} na wyjściu sieci pojawi się błąd, tzn.

$$d(\mathbf{x}) = -\text{sgn}(\mathbf{w} \circ \mathbf{x}). \quad (6.16)$$

Oczywiście

$$\|\mathbf{w}'\|^2 = \|\mathbf{w} + d(\mathbf{x})\mathbf{x}\|^2 = \|\mathbf{w}\|^2 + 2d(\mathbf{x})\mathbf{w} \circ \mathbf{x} + \|\mathbf{x}\|^2. \quad (6.17)$$

Wykorzystując zależności (6.16) i (6.17) oraz zakładając ograniczoność sygnałów wejściowych, mamy

$$\|\mathbf{w}'\|^2 < \|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 = \|\mathbf{w}\|^2 + C. \quad (6.18)$$

Po t krokach modyfikacji wag sieci zależności (6.15) oraz (6.18) przybierają postać

$$\mathbf{w}(t) \circ \mathbf{w}^* > \mathbf{w} \circ \mathbf{w}^* + t\delta \quad (6.19)$$

oraz

$$\|\mathbf{w}(t)\|^2 < \|\mathbf{w}\|^2 + tC. \quad (6.20)$$

Korzystając ze wzorów (6.10), (6.19) i (6.20), otrzymujemy

$$\cos \alpha(t) = \frac{\mathbf{w}^* \circ \mathbf{w}(t)}{\|\mathbf{w}(t)\|} > \frac{\mathbf{w}^* \circ \mathbf{w} + t\delta}{\sqrt{\|\mathbf{w}\|^2 + tC}}. \quad (6.21)$$

Dlatego też musi istnieć takie $t = t_{\max}$, dla którego $\cos(\alpha) = 1$. A więc istnieje skończona liczba kroków modyfikacji wag, po których wektor wag początkowych będzie realizował odwzorowanie $y = d(\mathbf{x})$. Jeżeli przyjmiemy, że wartości startowe wag są równe 0, to

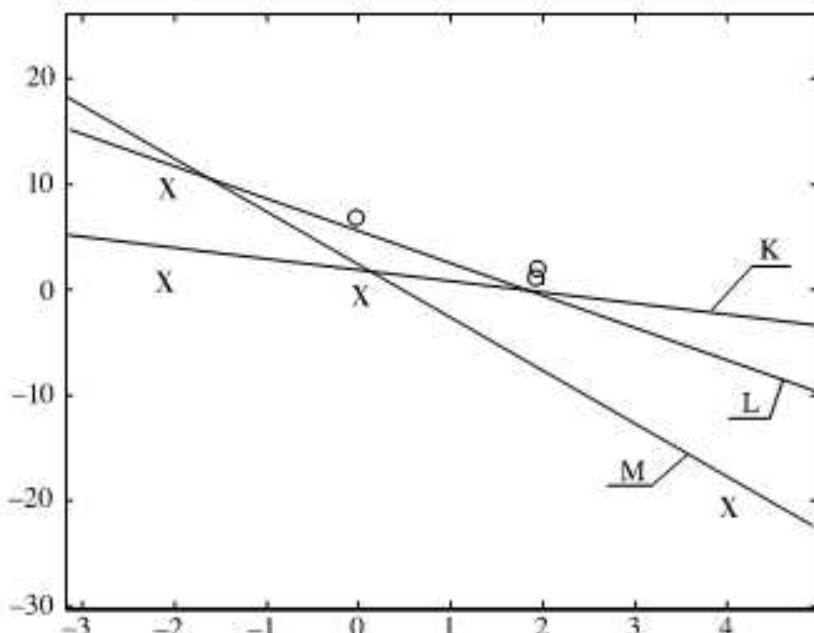
$$t_{\max} = \frac{C}{\delta^2}. \quad (6.22)$$

Przykład 6.1

Przedstawimy teraz przykład uczenia perceptronu. Omawiając jego działanie, stwierdziliśmy, iż ten model neuronu o dwóch wejściach dzieli płaszczyznę na dwie części (por. (6.5)). Wobec tego, jeżeli na płaszczyźnie umieścimy dwie klasy próbek, które będzie można rozdzielić za pomocą prostej, to perceptron w procesie uczenia będzie w stanie znaleźć tę linię podziału. W naszym doświadczeniu wykreślimy prostą wzorcową, oznaczoną literą L na rysunku 6.7. Przyjmiemy, że wszystkie punkty płaszczyzny leżące nad tą prostą reprezentują próbki z klasy 1, natomiast punkty leżące pod prostą L reprezentują klasę 2. Takich punktów na obu półpłaszczyznach znajduje się nieskończoność wiele i dlatego musimy wybrać po kilka próbek z każdej klasy. Chcemy, aby perceptron po nauczeniu dla próbek z klasy pierwszej na wyjściu podawał sygnał równy 1, natomiast dla próbek z klasy drugiej sygnał równy -1. W ten sposób zbudowaliśmy ciąg uczący przedstawiony w tabeli 6.1.

Tabela 6.1. Ciąg uczący w przykładzie 6.1

x_1	x_2	$d(\mathbf{x})$
2	1	1
2	2	1
0	6	1
-2	10	-1
-2	0	-1
0	0	-1
4	-20	-1



Rys. 6.7. Granice decyzyjne w przykładzie 6.1

Przyjmujemy następujące wartości początkowe wag perceptronu: $w_1 = 2$, $w_2 = 2$, $\theta = -4$. Na podstawie tych parametrów oraz wcześniejszych informacji wykreślamy prostą K, która pokazuje podział przestrzeni (granica decyzyjna), jaki wyznacza perceptron przed rozpoczęciem procesu uczenia. Perceptron po dziesięciu epokach algorytmu uczenia (10 razy podawaliśmy na wejścia neuronu wszystkie elementy ciągu uczącego) zaczął poprawnie klasyfikować wektory ciągu uczącego. Wagi jego przyjęły następujące wartości: $w_1 = 4$, $w_2 = 1$, $\theta = -1$, co odzwierciedla prostą M będącą granicą decyzyjną. Na rysunku 6.7 widzimy, że perceptron po nauczeniu poprawnie klasyfikuje próbki uczące, chociaż prosta M nie pokryła się z prostą wzorcową L.

6.2.3. Model Adaline

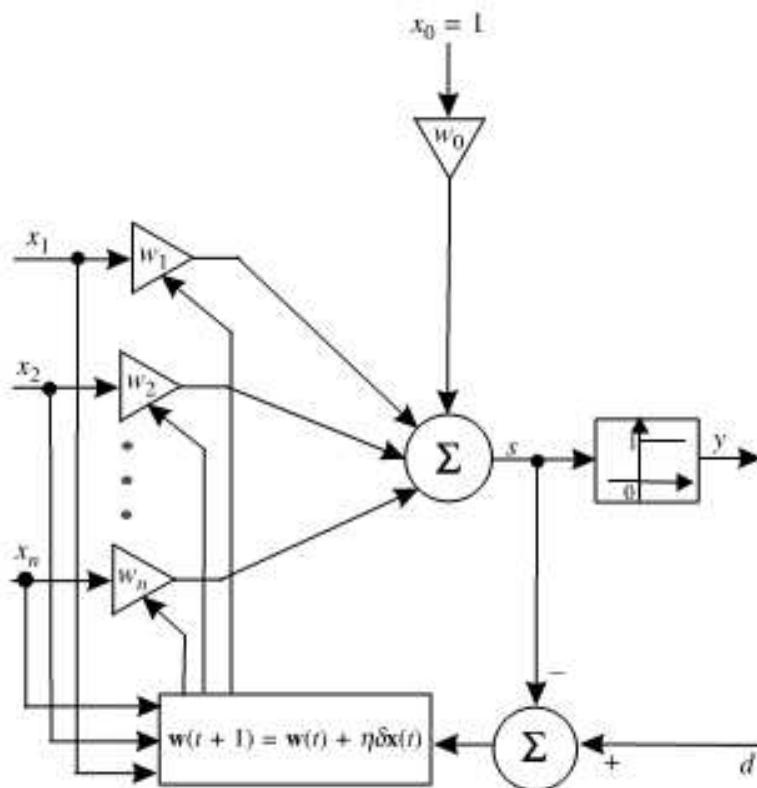
Schemat neuronu Adaline (ang. *Adaptive Linear Neuron*) przedstawiono na rysunku 6.8. Budowa tego neuronu jest bardzo podobna do modelu perceptronu, a jedyna różnica dotyczy algorytmu uczenia. Sposób wyznaczania sygnału wyjściowego jest identyczny z przedstawionym w poprzednim punkcie dotyczącym perceptronu. Jednak w przypadku neuronu typu Adaline porównuje się sygnał wzorcowy d z sygnałem s na wyjściu części liniowej neuronu (sumator). Stąd pochodzi nazwa tego typu neuronów. W ten sposób otrzymujemy błąd dany wzorem

$$\varepsilon = d - s. \quad (6.23)$$

Uczenie neuronu, czyli dobór wag, sprowadza się do minimalizacji funkcji określonej w sposób następujący:

$$Q(\mathbf{w}) = \frac{1}{2} \varepsilon^2 = \frac{1}{2} \left[d - \left(\sum_{i=0}^n w_i x_i \right) \right]^2. \quad (6.24)$$

Miarę błędu (6.24) określa się mianem *błędu średniego kwadratowego*. Uwzględniając tylko część liniową neuronu, możemy do modyfikacji wag użyć algorytmów gra-



Rys. 6.8. Schemat neuronu Adaline

dientowych, gdyż funkcja celu zdefiniowana zależnością (6.24) jest różniczkowalna. Do minimalizacji tejże funkcji użyjemy metody największego spadku. Metoda ta zostanie dokładniej omówiona przy okazji opisywania algorytmu wstecznej propagacji błędów. Wagi w neuronie typu Adaline modyfikuje się zgodnie ze wzorem

$$w_i(t+1) = w_i(t) - \eta \frac{\partial Q(w_i)}{\partial w_i}, \quad (6.25)$$

w którym η jest współczynnikiem uczenia. Zauważmy, że

$$\frac{\partial Q(w_i)}{\partial w_i} = \frac{\partial Q(w_i)}{\partial s} \cdot \frac{\partial s}{\partial w_i}. \quad (6.26)$$

Ponieważ s jest funkcją liniową względem wektora wag, więc możemy napisać

$$\frac{\partial s}{\partial w_i} = x_i. \quad (6.27)$$

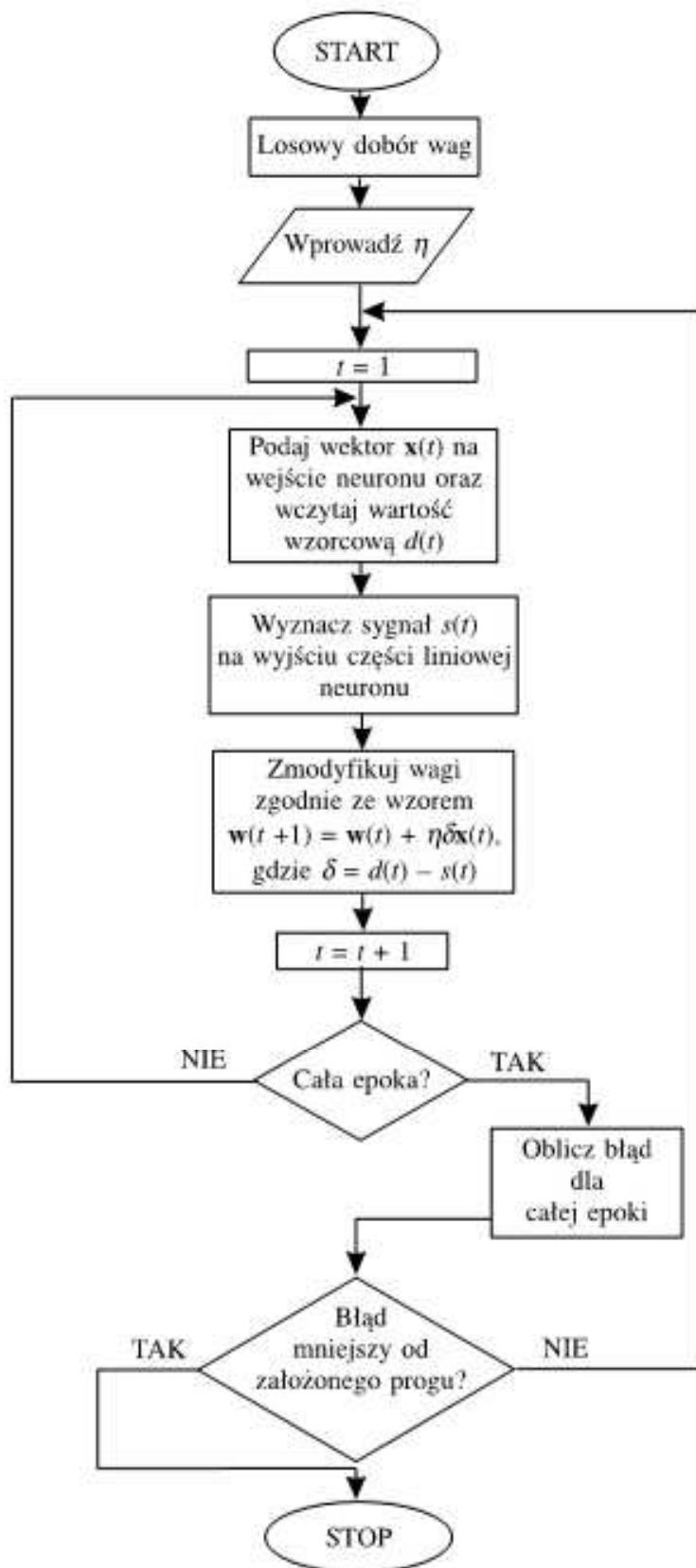
Ponadto

$$\frac{\partial Q(w_i)}{\partial s} = -(d - s). \quad (6.28)$$

Zatem zależność (6.25) przybiera postać

$$w_i(t+1) = w_i(t) + \eta \delta x_i, \quad (6.29)$$

gdzie $\delta = d - s$. Powyższa reguła nosi nazwę *reguły delta* (jest to szczególna postać tej reguły, ponieważ nie uwzględnia ona funkcji aktywacji neuronu). Na rysunku 6.9 przedstawiono schemat blokowy algorytmu uczenia neuronu typu Adaline za pomocą tej reguły.



Rys. 6.9. Schemat blokowy algorytmu uczenia neuronu typu Adaline

Neurony typu Adaline można również uczyć za pomocą rekurencyjnej metody najmniejszych kwadratów (ang. *Recursive Least Squares* — RLS). Jako miarę błędu przyjmuje się następujące wyrażenie:

$$Q(t) = \sum_{k=1}^t \lambda^{t-k} \varepsilon^2(k) = \sum_{k=1}^t \lambda^{t-k} [d(k) - \mathbf{x}^T(k) \mathbf{w}(t)]^2, \quad (6.30)$$

w którym λ jest współczynnikiem zapominania (ang. *forgetting factor*) wybieranym z przedziału $[0, 1]$. Zauważmy, że poprzednie błędy mają mniejszy wpływ na wartość wyrażenia (6.30). Obliczając gradient miary błędu, otrzymujemy następującą zależność:

$$\begin{aligned}\frac{\partial Q(t)}{\partial \mathbf{w}(t)} &= \frac{\partial \sum_{k=1}^t \lambda^{t-k} \varepsilon^2}{\partial \mathbf{w}(t)} \\ &= \frac{\partial \sum_{k=1}^t \lambda^{t-k} [d(k) - \mathbf{x}^T(k) \mathbf{w}(t)]^2}{\partial \mathbf{w}(t)} \\ &= -2 \sum_{k=1}^t \lambda^{t-k} [d(k) - \mathbf{x}^T(k) \mathbf{w}(t)] \mathbf{x}(k).\end{aligned}\quad (6.31)$$

Optymalne wartości wag powinny spełniać tzw. *równanie normalne*

$$\sum_{k=1}^t \lambda^{t-k} [d(k) - \mathbf{x}^T(k) \mathbf{w}(t)] \mathbf{x}(k) = \mathbf{0}. \quad (6.32)$$

Równanie (6.32) można przedstawić w postaci

$$\mathbf{r}(t) = \mathbf{R}(t) \mathbf{w}(t), \quad (6.33)$$

gdzie

$$\mathbf{R}(t) = \sum_{k=1}^t \lambda^{t-k} \mathbf{x}(k) \mathbf{x}^T(k) \quad (6.34)$$

jest $n \times n$ -wymiarową macierzą autokorelacji, oraz

$$\mathbf{r}(t) = \sum_{k=1}^t \lambda^{t-k} d(k) \mathbf{x}(k) \quad (6.35)$$

jest $n \times 1$ -wymiarowym wektorem korelacji wzajemnej sygnału wejściowego i sygnału wzorcowego. Zakładamy, że sygnały te są realizacjami stacjonarnych procesów stochastycznych. Rozwiążanie równania normalnego (6.33) przybiera postać

$$\mathbf{w}(t) = \mathbf{R}^{-1}(t) \mathbf{r}(t), \quad (6.36)$$

jeżeli $\det \mathbf{R}(t) \neq 0$. Zastosujemy teraz algorytm RLS w celu uniknięcia operacji odwracania macierzy w równaniu (6.36) i rozwiążemy równanie normalne (6.33) w sposób rekurencyjny.

Zauważmy, że macierz $\mathbf{R}(t)$ oraz wektor $\mathbf{r}(t)$ można przedstawić w postaci

$$\mathbf{R}(t) = \lambda \mathbf{R}(t-1) + \mathbf{x}(t) \mathbf{x}^T(t) \quad (6.37)$$

oraz

$$\mathbf{r}(t) = \lambda \mathbf{r}(t-1) + \mathbf{x}(t) d(t). \quad (6.38)$$

Zastosujemy teraz lemat o odwrotności macierzy. Niech \mathbf{A} i \mathbf{B} będą dodatnio określonymi $n \times n$ -wymiarowymi macierzami takimi, że

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C} \mathbf{D}^{-1} \mathbf{C}^T, \quad (6.39)$$

gdzie \mathbf{D} jest dodatnio określona $m \times m$ -wymiarową macierzą, natomiast \mathbf{C} jest $n \times m$ -wymiarową macierzą. Wówczas

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{BC}(\mathbf{D} + \mathbf{C}^T \mathbf{BC})^{-1} \mathbf{C}^T \mathbf{B}. \quad (6.40)$$

Porównując wzory (6.40) i (6.37), otrzymujemy

$$\begin{aligned} \mathbf{A} &= \mathbf{R}(t), \\ \mathbf{B}^{-1} &= \lambda \mathbf{R}(t-1), \\ \mathbf{C} &= \mathbf{x}(t), \\ \mathbf{D} &= 1. \end{aligned} \quad (6.41)$$

Zatem

$$\mathbf{P}(t) = \lambda^{-1} [\mathbf{I} - \mathbf{g}(t) \mathbf{x}^T(t)] \mathbf{P}(t-1), \quad (6.42)$$

gdzie

$$\mathbf{P}(t) = \mathbf{R}^{-1}(t) \quad (6.43)$$

oraz

$$\mathbf{g}(t) = \frac{\mathbf{P}(t-1) \mathbf{x}(t)}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)}. \quad (6.44)$$

Wykażemy prawdziwość następującego równania:

$$\mathbf{g}(t) = \mathbf{P}(t) \mathbf{x}(t). \quad (6.45)$$

W wyniku prostych operacji algebraicznych otrzymujemy

$$\begin{aligned} \mathbf{g}(t) &= \frac{\mathbf{P}(t-1) \mathbf{x}(t)}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \\ &= \frac{\lambda^{-1} [\lambda \mathbf{P}(t-1) \mathbf{x}(t) + \mathbf{P}(t-1) \mathbf{x}(t) \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)]}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \\ &\quad - \frac{\lambda^{-1} [\mathbf{P}(t-1) \mathbf{x}(t) \mathbf{x}^T(t) + \mathbf{P}(t-1) \mathbf{x}(t)]}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \\ &= \frac{\lambda^{-1} [(\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)) \mathbf{I}] \mathbf{P}(t-1) \mathbf{x}(t)}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \\ &\quad - \frac{\lambda^{-1} [\mathbf{P}(t-1) \mathbf{x}(t) \mathbf{x}^T(t)] \mathbf{P}(t-1) \mathbf{x}(t)}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \\ &= \lambda^{-1} \left[\mathbf{I} - \frac{\mathbf{P}(t-1) \mathbf{x}(t) \mathbf{x}^T(t)}{\lambda + \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)} \right] \mathbf{P}(t-1) \mathbf{x}(t) \\ &= \lambda^{-1} [\mathbf{I} - \mathbf{g}(t) \mathbf{x}^T(t)] \mathbf{P}(t-1) \mathbf{x}(t) = \mathbf{P}(t) \mathbf{x}(t). \end{aligned} \quad (6.46)$$

Z zależności (6.38) oraz (6.36) wynika, że

$$\mathbf{w}(t) = \mathbf{R}^{-1}(t) \mathbf{r}(t) = \lambda \mathbf{P}(t) \mathbf{r}(t-1) + \mathbf{P}(t) \mathbf{x}(t) \mathbf{d}(t). \quad (6.47)$$

Z równania (6.42) oraz (6.47) otrzymujemy

$$\mathbf{w}(t) = [\mathbf{I} - \mathbf{g}(t) \mathbf{x}^T(t)] \mathbf{P}(t-1) \mathbf{r}(t-1) + \mathbf{P}(t) \mathbf{x}(t) \mathbf{d}(t). \quad (6.48)$$

Konsekwencją zależności (6.38) i (6.36) jest następujący związek:

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \mathbf{g}(t)\mathbf{x}^T(t)\mathbf{w}(t-1) + \mathbf{P}(t)\mathbf{x}(t)d(t). \quad (6.49)$$

Uwzględniając związek (6.45) w zależności (6.49), otrzymujemy następującą rekursję:

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{g}(t)[d(t) - \mathbf{x}^T(t)\mathbf{w}(t-1)]. \quad (6.50)$$

W konsekwencji algorytm RLS zastosowany do uczenia neuronu typu Adaline przybiera następującą postać:

$$\varepsilon(t) = d(t) - \mathbf{x}^T(t)\mathbf{w}(t-1) = d(t) - y(t), \quad (6.51)$$

$$\mathbf{g}(t) = \frac{\mathbf{P}(t-1)\mathbf{x}(t)}{\lambda + \mathbf{x}^T(t)\mathbf{P}(t-1)\mathbf{x}(t)}, \quad (6.52)$$

$$\mathbf{P}(t) = \lambda^{-1}[\mathbf{I} - \mathbf{g}(t)\mathbf{x}^T(t)]\mathbf{P}(t-1), \quad (6.53)$$

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{g}(t)\varepsilon(t). \quad (6.54)$$

Jako wartości początkowe zazwyczaj przyjmuje się

$$\mathbf{P}(0) = \gamma \mathbf{I}, \quad \gamma > 0, \quad (6.55)$$

gdzie γ jest stałą, natomiast \mathbf{I} jest macierzą jednostkową.

6.2.4. Model neuronu sigmoidalnego

Budowa neuronu sigmoidalnego jest analogiczna do dwóch ostatnio omówionych modeli, tzn. do perceptronu i neuronu typu Adaline. Nazwa pochodzi od funkcji aktywacji, która przybiera postać funkcji sigmoidalnej unipolarnej lub bipolarnej. Są to funkcje ciągłe i wyrażają się następującymi zależnościami:

$$f(x) = \frac{1}{1 + e^{-\beta x}} \text{ — funkcja unipolarna}$$

oraz

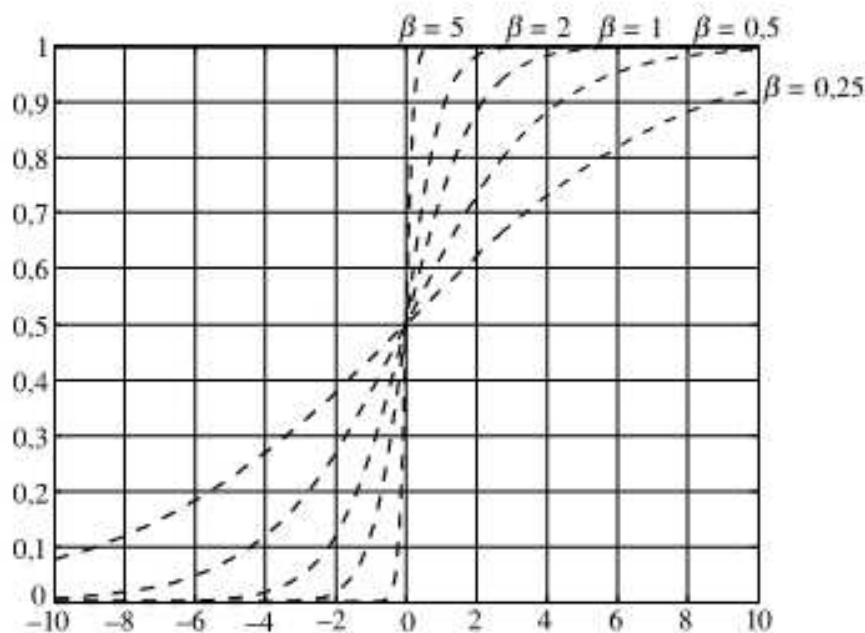
$$f(x) = \tanh(\beta x) = \frac{1 - e^{\beta x}}{1 + e^{\beta x}} \text{ — funkcja bipolarna.}$$

Na rysunku 6.10 przedstawiono przebiegi funkcji unipolarnej dla różnych wartości parametru β . Czytelnik może zauważyć, iż przy małej wartości współczynnika β funkcja ma kształt łagodny, wraz ze wzrostem współczynnika wykres staje się bardziej stromy, aż wreszcie funkcja ma charakter progowy. Cechą, która niewątpliwie jest ogromną zaletą neuronów sigmoidalnych, jest różniczkowalność funkcji aktywacji. Ponadto pochodne tych funkcji można łatwo obliczyć, gdyż przybierają one następującą postać:

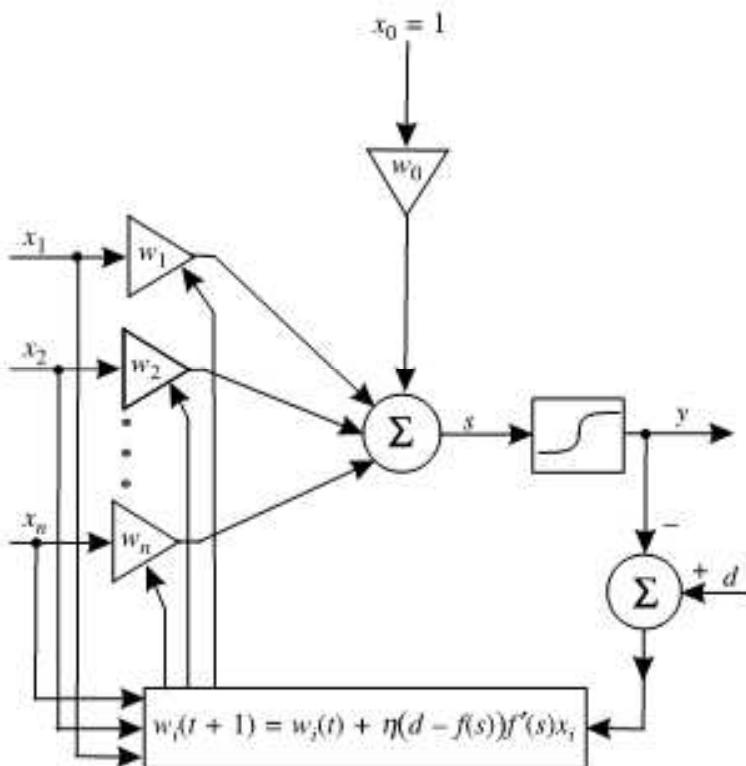
$$f'(x) = \beta f(x)(1 - f(x)) \text{ dla funkcji unipolarnej,} \quad (6.56)$$

$$f'(x) = \beta(1 - f^2(x)) \text{ dla funkcji bipolarnej.} \quad (6.57)$$

Schemat neuronu sigmoidalnego przedstawiono na rysunku 6.11.



Rys. 6.10. Wykresy unipolarnej funkcji aktywacji dla różnych wartości parametru β



Rys. 6.11. Schemat neuronu sigmoidalnego

Sygnał wyjściowy jest dany wzorem

$$y(t) = f\left(\sum_{i=0}^n w_i(t)x_i(t)\right). \quad (6.58)$$

Miarę błędu Q definiuje się jako kwadrat różnicy wartości wzorcowej i wartości otrzymanej na wyjściu neuronu, tzn.

$$Q(\mathbf{w}) = \frac{1}{2} \left[d - f\left(\sum_{i=0}^n w_i x_i\right) \right]^2. \quad (6.59)$$

Podobnie jak w przypadku neuronu typu Adaline do uczenia używa się reguły największego spadku, jednak teraz w wyprowadzeniach uwzględnimy funkcję aktywacji. Wagi neuronu uaktualniamy zgodnie ze wzorem

$$w_i(t+1) = w_i(t) - \eta \frac{\partial Q(w_i)}{\partial w_i}. \quad (6.60)$$

Wyznaczamy pochodną miary błędu względem wag. Oczywiście

$$\frac{\partial Q(w_i)}{\partial w_i} = \frac{\partial Q(w_i)}{\partial s} \cdot \frac{\partial s}{\partial w_i} \quad (6.61)$$

oraz

$$\frac{\partial s}{\partial w_i} = x_i. \quad (6.62)$$

Stąd

$$\frac{\partial Q(w_i)}{\partial w_i} = \frac{\partial Q(w_i)}{\partial s} \cdot x_i. \quad (6.63)$$

Łatwo zauważyc, że

$$\frac{\partial Q(w_i)}{\partial s} = -(d - f(s)) \cdot f'(s). \quad (6.64)$$

Oznaczmy

$$\delta = -(d - f(s)) \cdot f'(s). \quad (6.65)$$

Zgodnie ze wzorami (6.60) i (6.65) modyfikacji wag w kroku $t+1$ dokonujemy w następujący sposób:

$$w_i(t+1) = w_i(t) - \eta \delta x_i = w_i(t) + \eta (d - f(s)) f'(s) x_i. \quad (6.66)$$

Przedstawimy teraz alternatywny sposób uczenia neuronu sigmoidalnego z wykorzystaniem algorytmu RLS. Rozważymy dwa przypadki różniące się sposobem definiowania błędu. W przypadku pierwszym sygnał błędu jest wyznaczany na wyjściu części liniowej neuronu. Zatem miara błędu jest w postaci

$$\begin{aligned} Q(t) &= \sum_{k=1}^t \lambda^{t-k} e^2(k) \\ &= \sum_{k=1}^t \lambda^{t-k} [b(k) - \mathbf{x}^T(k) \mathbf{w}(t)]^2, \end{aligned} \quad (6.67)$$

gdzie

$$b(k) = f^{-1}(d(k)) = \begin{cases} \ln \frac{d(k)}{1-d(k)} & \text{w przypadku funkcji unipolarnej,} \\ \frac{1}{2} \ln \frac{1+d(k)}{1-d(k)} & \text{w przypadku funkcji bipolarnej} \end{cases} \quad (6.68)$$

ma interpretację sygnału zadanego na wyjściu części liniowej neuronu. Obecnie równanie normalne przybiera postać

$$\frac{\partial Q(t)}{\partial \mathbf{w}(t)} = -2 \sum_{k=1}^t \lambda^{t-k} [b(k) - \mathbf{x}^T(k) \mathbf{w}(t)] \mathbf{x}^T(k) \quad (6.69)$$

lub w postaci wektorowej

$$\mathbf{r}(t) = \mathbf{R}(t)\mathbf{w}(t), \quad (6.70)$$

gdzie

$$\mathbf{R}(t) = \sum_{k=1}^t \lambda^{t-k} \mathbf{x}(k) \mathbf{x}^T(k) \quad (6.71)$$

oraz

$$\mathbf{r}(t) = \sum_{k=1}^t \lambda^{t-k} b(k) \mathbf{x}(k). \quad (6.72)$$

Zauważmy, że równania (6.71) i (6.72) są analogiczne do równań (6.34) i (6.35). Zatem algorytm RLS przybiera następującą postać:

$$e(t) = b(t) - \mathbf{x}^T(t)\mathbf{w}(t-1) = b(t) - s(t), \quad (6.73)$$

$$\mathbf{g}(t) = \frac{\mathbf{P}(t-1)\mathbf{x}(t)}{\lambda + \mathbf{x}^T(t)\mathbf{P}(t-1)\mathbf{x}(t)}, \quad (6.74)$$

$$\mathbf{P}(t) = \lambda^{-1} [\mathbf{I} - \mathbf{g}(t)\mathbf{x}^T(t)]\mathbf{P}(t-1), \quad (6.75)$$

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{g}(t)e(t), \quad (6.76)$$

przy czym warunki początkowe określają wzór (6.55).

W przypadku drugim błąd jest określany na wyjściu części nieliniowej neuronu. Miara błędu przybiera postać

$$\begin{aligned} Q(t) &= \sum_{k=1}^t \lambda^{t-k} \varepsilon^2(k) \\ &= \sum_{k=1}^t \lambda^{t-k} [d(k) - f(\mathbf{x}^T(k)\mathbf{w}(t))]^2. \end{aligned} \quad (6.77)$$

Wyznaczając pochodną cząstkową miary (6.77) względem wektora $\mathbf{w}(t)$ i przyrównując wynik do $\mathbf{0}$, mamy

$$\begin{aligned} \frac{\partial Q(t)}{\partial \mathbf{w}(t)} &= 2 \sum_{k=1}^t \lambda^{t-k} \frac{\partial \varepsilon(k)}{\partial \mathbf{w}(t)} \varepsilon(k) \\ &= -2 \sum_{k=1}^t \lambda^{t-k} \frac{\partial y(k)}{\partial s(k)} \frac{\partial s(k)}{\partial \mathbf{w}(t)} \varepsilon(k) = \mathbf{0}. \end{aligned} \quad (6.78)$$

W toku dalszych obliczeń otrzymujemy

$$\begin{aligned} \sum_{k=1}^t \lambda^{t-k} \frac{\partial y(k)}{\partial s(k)} \frac{\partial s(k)}{\partial \mathbf{w}(t)} [d(k) - y(k)] \\ &= \sum_{k=1}^t \lambda^{t-k} \frac{\partial y(k)}{\partial s(k)} \mathbf{x}^T(k) [d(k) - y(k)] \\ &= \sum_{k=1}^t \lambda^{t-k} \frac{\partial y(k)}{\partial s(k)} \mathbf{x}^T(k) [f(b(k) - f(s(k)))] = \mathbf{0}. \end{aligned} \quad (6.79)$$

W wyniku zastosowania rozwinięcia Taylora do wyrażenia w nawiasie kwadratowym wzoru (6.79) dostajemy

$$f(b(k)) \approx f(s(k)) + f'(s(k))(b(k) - s(k)), \quad (6.80)$$

gdzie

$$b(t) = f^{-1}(d(t)). \quad (6.81)$$

W konsekwencji wzorów (6.79) i (6.80) otrzymujemy równanie

$$\sum_{k=1}^t \lambda^{t-k} f'^2(s(k)) [b(k) - \mathbf{x}^T(k) \mathbf{w}(t)] \mathbf{x}(k) = \mathbf{0}. \quad (6.82)$$

Równanie (6.82) w postaci wektorowej przybiera postać

$$\mathbf{r}(t) = \mathbf{R}(t) \mathbf{w}(t), \quad (6.83)$$

gdzie

$$\mathbf{R}(t) = \sum_{k=1}^t \lambda^{t-k} f'^2(s(k)) \mathbf{x}(k) \mathbf{x}^T(k) \quad (6.84)$$

oraz

$$\mathbf{r}(t) = \sum_{k=1}^t \lambda^{t-k} f'^2(s(k)) b(k) \mathbf{x}(k). \quad (6.85)$$

Stosując następujące podstawienia we wzorach (6.73)–(6.76):

$$\mathbf{x}(k) \rightarrow f'(s(k)) \mathbf{x}(k), \quad (6.86)$$

$$b(k) \rightarrow f'(s(k)) b(k), \quad (6.87)$$

otrzymujemy następującą postać algorytmu RLS zastosowanego do uczenia neuronu sigmoidalnego:

$$\varepsilon(t) = f'(s(t)) [b(t) - \mathbf{x}^T(t) \mathbf{w}(t-1)] \approx d(t) - y(t), \quad (6.88)$$

$$g(t) = \frac{f'(s(t)) \mathbf{P}(t-1) \mathbf{x}(t)}{\lambda + f'^2(s(t)) \mathbf{x}^T(t) \mathbf{P}(t-1) \mathbf{x}(t)}, \quad (6.89)$$

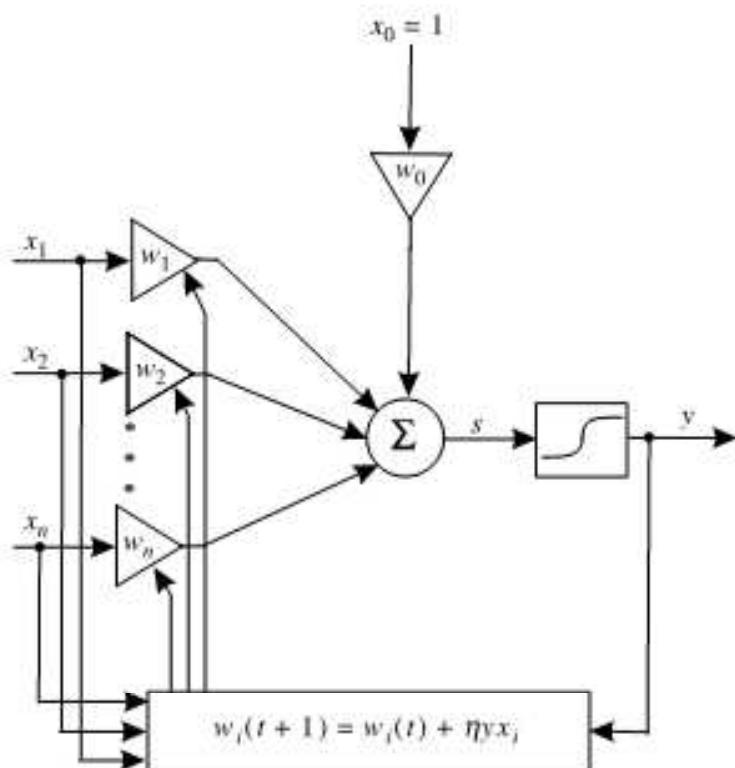
$$\mathbf{P}(t) = \lambda^{-1} [\mathbf{I} - f'(s(t)) \mathbf{g}(t) \mathbf{x}^T(t)] \mathbf{P}(t-1), \quad (6.90)$$

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{g}(t) \varepsilon(t). \quad (6.91)$$

Warunki początkowe określają zależność (6.55).

6.2.5. Model neuronu Hebb'a

Model neuronu Hebb'a został przedstawiony na rysunku 6.12. Jest to identyczna struktura jak w przypadku modelu typu Adaline oraz neuronu sigmoidalnego, ale charakteryzuje się specyficzną metodą uczenia, znaną pod nazwą *reguły Hebb'a*. Reguła ta występuje w wersji bez nauczyciela lub z nauczycielem. Hebb [74] zajmował się działaniem komórek nerwowych. Podczas swych badań zauważył, że połączenie pomiędzy dwiema komórkami jest wzmacnianie, jeżeli w tym samym czasie obie komórki stają się aktywne.



Rys. 6.12. Schemat neuronu Hebb'a

Postępując analogicznie, zaproponował on algorytm, zgodnie z którym modyfikację wag przeprowadza się następująco:

$$w_i(t+1) = w_i(t) + \Delta w_i, \quad (6.92)$$

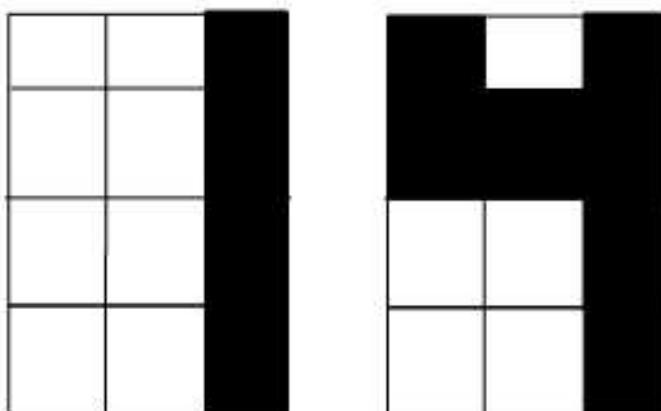
natomiast

$$\Delta w_i = \eta y x_i. \quad (6.93)$$

W przypadku pojedynczego neuronu w trakcie uczenia będziemy modyfikować wartość wagi w_i proporcjonalnie zarówno do wartości sygnału podanego na i -te wejście, jak i sygnału wyjściowego y z uwzględnieniem współczynnika uczenia η . Zauważmy, że w tym przypadku nie podajemy wzorcowej wartości wyjściowej, stosujemy więc tu metodę uczenia bez nauczyciela. Niewielka modyfikacja zależności (6.93) prowadzi do drugiej metody uczenia neuronu Hebb'a — z nauczycielem

$$\Delta w_i = \eta x_i d, \quad (6.94)$$

gdzie d oznacza sygnał wzorcowy. Pewną wadą omawianego przez nas algorytmu jest to, że wartości wag mogą wzrastać do dowolnie dużych liczb. Dlatego w literaturze wprowadza się różne modyfikacje reguły Hebb'a.



Rys. 6.13. Ilustracja do przykładu 6.2

Przykład 6.2

Przedstawimy teraz przykład uczenia neuronu z wykorzystaniem reguły Hebla w wersji z nauczycielem. Naszym zadaniem będzie taka modyfikacja wag neuronu, aby rozpoznać cyfry 1 i 4, schematycznie pokazane na rysunku 6.13. Przyporządkowując białym polom na tym rysunku liczbę 1, a czarnym liczbę -1, otrzymujemy dwa wektory wchodzące w skład ciągu uczącego:

$$\begin{bmatrix} -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \text{ — dla cyfry 1,}$$

$$\begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \text{ — dla cyfry 4.}$$

Dla wzorca pierwszego (cyfra 1) będziemy żądać, aby na wyjściu neuronu pojawiał się sygnał $d = -1$, natomiast dla drugiego (cyfra 4) wartość wzorcową ustaliliśmy jako $d = 1$. Ponieważ znamy wzorce wejściowe i wyjściowe, więc wagi neuronu w kolejnych iteracjach algorytmu będą ulegały modyfikacji zgodnie z zależnością (6.94). Ich wartości początkowe są równe 0. Neuron z funkcją aktywacji typu signum uczyliśmy 100 epokami, współczynnik uczenia przyjęliśmy równy 0,2. Po nauczeniu neuronu i podaniu na jego wejście pierwszego wektora uczącego na wyjściu pojawił się sygnał $s = -120$, natomiast w przypadku drugiego wektora uczącego na wyjściu pojawił się sygnał $s = 120$. Można słusznie przypuszczać, iż wraz ze wzrostem liczby epok wartości te także będą wzrastały. Wektor wag po nauczeniu przybrał następującą postać: $w = [40 \ 0 \ 0 \ 40 \ 40 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$. Widzimy, że zmianom uległy tylko te składowe wektora wag, które odpowiadały różnicom pomiędzy poszczególnymi składowymi wektorów uczących.

6.3. Sieci jednokierunkowe wielowarstwowe

6.3.1. Budowa i działanie sieci

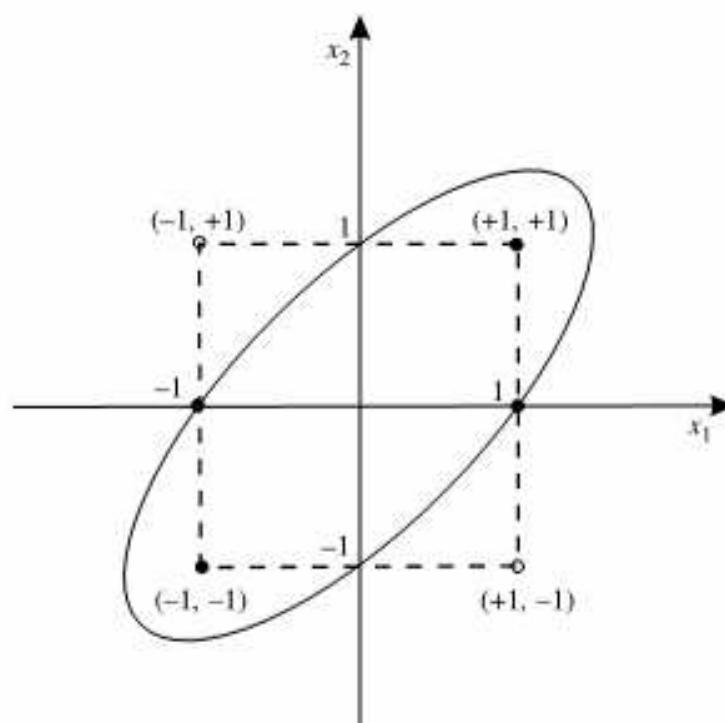
W poprzednim rozdziale omówiliśmy różne modele neuronów. Wykazaliśmy, iż neurony te mogą się uczyć, tzn. dostosowywać wartości swoich wag do wartości ciągu uczącego. Ponadto opisując perceptron, pokazaliśmy, że gdy ma on n wejść, wówczas dzieli n -wymiarową przestrzeń na dwie półprzestrzenie. Są one rozdzielone $n - 1$ -wymiarową

hiperplaszczyzną. Zakres możliwych problemów, które można rozwiązać, stosując pojedynczy perceptron, jest raczej wąski. Przykładem może być problem funkcji logicznej „wyłącznego lub” — XOR. Ciąg uczący dla tego problemu przedstawiono w tabeli 6.2.

Tabela 6.2. Ciąg uczący dla problemu XOR

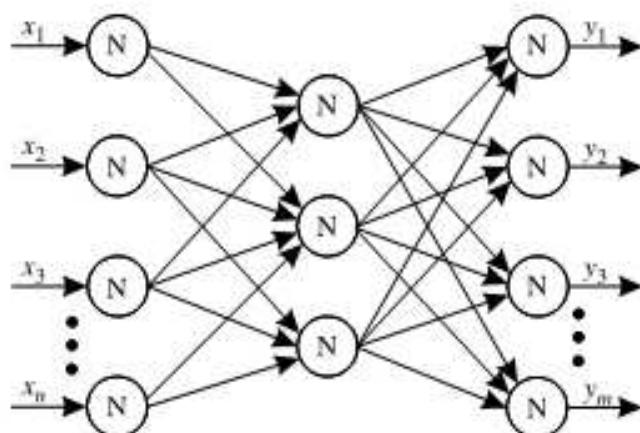
x_1	x_2	$d = \text{XOR}(x_1, x_2)$
+1	+1	-1
+1	-1	1
-1	+1	1
-1	-1	-1

Na rysunku 6.14 zaznaczono wartości ciągu uczącego z tabeli 6.2. Z rysunku wynika, że nie istnieje prosta, która rozdzielałaby punkty o wartościach funkcji XOR różnych -1 od punktów o wartościach równych 1. W tym przypadku funkcję przykładowej granicy decyzyjnej pełni elipsa, a zatem algorytm przedstawiony na rysunku 6.5 nie byłby zbieżny. Nie jesteśmy w stanie tak dobrą wagą pojedynczego perceptronu, aby rozwiązać problem XOR. Z pomocą przychodzą nam sieci wielowarstwowe. Do problemu XOR wróćmy, gdy przedstawimy metody uczenia tychże sieci. Czym właściwie są sieci wielowarstwowe? To nic innego jak odpowiednio połączone ze sobą neurony rozmieszczone w dwóch lub więcej warstwach. Z reguły są to neurony sigmoidalne, ale używa się również neuronów o innych funkcjach aktywacji, np. liniowych — najczęściej stosowanych w ostatnich warstwach struktury sieci neuronowej. W wielowarstwowych sieciach neuronowych muszą istnieć co najmniej dwie warstwy: wejściowa i wyjściowa. Między nimi natomiast mogą znajdować się warstwy ukryte. Jeżeli sieć zawiera tylko dwie warstwy, to warstwę wejściową utożsamiamy z warstwą ukrytą. W niektórych opracowaniach przez warstwę wejściową rozumie się wektor sygnałów wejściowych podawanych na sieć neuronową. W rozważanych przez nas strukturach neurony przekazują sygnały tylko między



Rys. 6.14. Ilustracja zagadnienia XOR

różnymi warstwami. W obrębie tej samej warstwy neurony nie mogą się łączyć ze sobą. Sygnały są przekazywane od warstwy wejściowej do wyjściowej (stąd nazwa jednokierunkowe), przy czym nie występują sprzężenia zwrotne do warstw poprzednich. Typowy schemat sieci jednokierunkowej trójwarstwowej przedstawiono na rysunku 6.15.



Rys. 6.15. Schemat trójwarstwowej sieci neuronowej

W kolejnych punktach pokażemy różne algorytmy uczenia sieci wielowarstwowych z nauczycielem. Na samym początku przedstawimy działanie algorytmu wstecznej propagacji błędów (ang. *error backpropagation*) oraz kilka jego modyfikacji. Kolejnym algorytmem, jaki zostanie omówiony w tym rozdziale, będzie algorytm RLS. Następnie pokażemy, jaki wpływ ma dobór odpowiedniej struktury sieci na przebieg uczenia i działania sieci. Jak wspomnieliśmy wcześniej, przedstawione w tym rozdziale algorytmy należą do algorytmów uczenia sieci z nauczycielem, zwanego inaczej *uczeniem nadzorowanym*. Przypomnijmy, co oznacza to pojęcie. W przypadku tych algorytmów zakładamy, iż w ciągu uczącym znajdują się następujące pary: wektor wartości wejściowych oraz wektor wzorcowych sygnałów wyjściowych. Uczenie będzie więc przebiegać w sposób następujący: najpierw podajemy wartości wejściowe z ciągu uczącego na wejście sieci, a następnie obliczamy wartości wyjściowe kolejno dla wszystkich neuronów od warstwy wejściowej do warstwy wyjściowej. W ten sposób poznamy odpowiedź sieci na sygnał (wzorzec) podany na jej wejście. Ponieważ wiemy, jaka powinna być wartość wyjściowa (znajduje się ona w ciągu uczącym), będziemy starali się tak zmodyfikować wagę w sieci, aby wartość otrzymaną na wyjściu zbliżyć do wartości wzorcowej. Stąd też nazwa tego typu algorytmów (z nauczycielem lub uczenie nadzorowane), gdyż „nauczyciel” wskazuje, jaka powinna być odpowiedź sieci.

6.3.2. Algorytm wstecznej propagacji błędów

Podczas przedstawiania różnych modeli neuronów omówiliśmy podstawowe techniki ich uczenia. Najczęściej odbywało się to w sposób następujący. Obliczaliśmy sumę wartości iloczynów sygnałów wejściowych i odpowiadających im wag. Następnie poddawaliśmy otrzymaną w ten sposób wartość działaniu odpowiednio zdefiniowanej funkcji aktywacji i otrzymywaliśmy wartość wyjściową neuronu. Ponieważ znaliśmy wartość wyjściową, jaka powinna być na wyjściu (wartość wyjściowa wzorcowa otrzymana z ciągu uczącego), w łatwy sposób mogliśmy zdefiniować błąd na wyjściu neuronu. Błędem

tym była różnica między wartością otrzymaną na jego wyjściu a wartością wzorcową. W identyczny sposób możemy zdefiniować błąd dla warstwy ostatniej w przypadku sieci wielowarstwowych. W tym momencie pojawia się jednak problem zdefiniowania błędu dla warstw ukrytych, ponieważ nauczyciel nie wie, jaka powinna być wartość na wyjściu poszczególnych neuronów warstw ukrytych. Z pomocą przychodzi nam tu najbardziej rozpowszechniona technika uczenia sieci neuronowych wielowarstwowych zwana *metodą wstecznej propagacji błędów*. Aby wyprowadzić ten algorytm, musimy odpowiednio zdefiniować miarę błędu. Będzie to funkcja, w której w rolach zmiennych występują wszystkie wagi wielowarstwowej sieci neuronowej. Oznaczmy tę funkcję przez $Q(\mathbf{w})$, gdzie \mathbf{w} oznacza wektor wszystkich wag sieci. Będziemy dążyć w trakcie uczenia sieci do znalezienia minimum funkcji Q względem wektora \mathbf{w} . Rozwińmy zatem rozważaną przez nas funkcję w szereg Taylora w najbliższym sąsiedztwie znanego aktualnego rozwiązania \mathbf{w} . Rozwinięcie to przedstawimy wzdłuż kierunku \mathbf{p} w sposób następujący:

$$Q(\mathbf{w} + \mathbf{p}) = Q(\mathbf{w}) + [\mathbf{g}(\mathbf{w})]^T \mathbf{p} + 0,5 \mathbf{p}^T \mathbf{H}(\mathbf{w}) \mathbf{p} + \dots, \quad (6.95)$$

gdzie $\mathbf{g}(\mathbf{w})$ oznacza wektor gradientu, tzn.

$$\mathbf{g}(\mathbf{w}) = \left[\frac{\partial Q}{\partial w_1}, \frac{\partial Q}{\partial w_2}, \frac{\partial Q}{\partial w_3}, \dots, \frac{\partial Q}{\partial w_n} \right]^T, \quad (6.96)$$

natomiast $\mathbf{H}(\mathbf{w})$ jest hesjanem, tzn. macierzą drugich pochodnych

$$\mathbf{H}(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 Q}{\partial w_1 \partial w_1} & \dots & \frac{\partial^2 Q}{\partial w_1 \partial w_n} \\ \vdots & & \vdots \\ \frac{\partial^2 Q}{\partial w_n \partial w_1} & \dots & \frac{\partial^2 Q}{\partial w_n \partial w_n} \end{bmatrix}, \quad (6.97)$$

Modyfikację wag przeprowadza się w następujący sposób:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta(t) \mathbf{p}(t), \quad (6.98)$$

gdzie η jest współczynnikiem uczenia (sposób dobierania tego parametru opisany jest w dalszej części rozdziału). Modyfikację wag można wykonywać tak długo, aż funkcja Q osiągnie minimum lub jej wartość spadnie poniżej założonego progu. Zadanie sprowadza się więc do wyznaczania takiego wektora kierunkowego \mathbf{p} , aby w kolejnych krokach algorytmu błąd na wyjściu sieci malał. Oznacza to, iż wymagamy spełnienia nierówności $Q(\mathbf{w}(t+1)) < Q(\mathbf{w}(t))$ w kolejnych krokach iteracji. Ograniczmy szereg Taylora aproksymującą funkcję błędu Q do rozwinięcia liniowego, tzn.

$$Q(\mathbf{w} + \mathbf{p}) = Q(\mathbf{w}) + [\mathbf{g}(\mathbf{w})]^T \mathbf{p}. \quad (6.99)$$

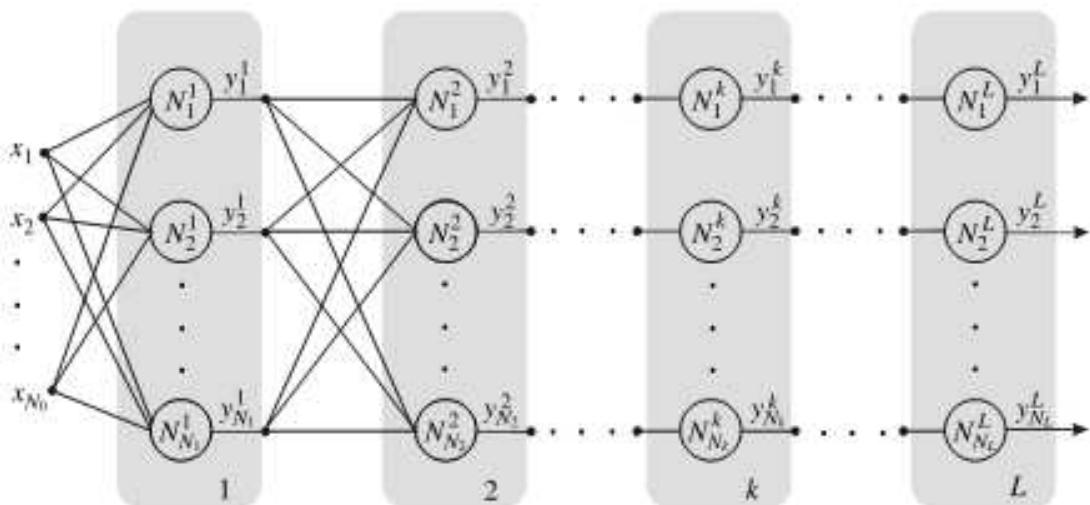
Ponieważ funkcja $Q(\mathbf{w})$ zależy od wag wyznaczonych w kroku t , natomiast $Q(\mathbf{w} + \mathbf{p})$ zależy od wag wyznaczonych w kroku $t + 1$, więc aby zachodziła zależność $Q(\mathbf{w}(t+1)) < Q(\mathbf{w}(t))$, wystarczy tak dobrać wektor $\mathbf{p}(t)$, żeby $\mathbf{g}(\mathbf{w}(t))^T \mathbf{p}(t) < 0$. Łatwo zauważać, że warunek ten jest spełniony, jeśli przyjmiemy

$$\mathbf{p}(t) = -\mathbf{g}(\mathbf{w}(t)). \quad (6.100)$$

Podstawiając zależność (6.100) do wzoru (6.98), otrzymujemy następujący wzór określający sposób zmiany wag wielowarstwowej sieci neuronowej

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \mathbf{g}(\mathbf{w}(t)). \quad (6.101)$$

Zależność (6.101) znana jest w literaturze pod nazwą *reguły największego spadku*. Abyśmy mogli efektywnie wykorzystać zależność (6.101) do wyprowadzenia algorytmu wstecznej propagacji błędów, musimy formalnie opisać schemat wielowarstwowej sieci neuronowej oraz wprowadzić odpowiednie oznaczenia.



Rys. 6.16. Wielowarstwowa sieć neuronowa

Schemat taki przedstawiono na rysunku 6.16. W każdej warstwie znajduje się N_k elementów, $k = 1, \dots, L$, oznaczonych jako N_i^k , $i = 1, \dots, N_k$. Elementy N_i^k będziemy nazywać neuronami, przy czym każdy z nich może być neuronem sigmoidalnym. Omawiana sieć neuronowa ma N_0 wejść, na które są podawane sygnały $x_1(t), \dots, x_{N_0}(t)$ zapisywane w postaci wektora

$$\mathbf{x} = [x_1(t), \dots, x_{N_0}(t)]^T \quad t = 1, 2, \dots \quad (6.102)$$

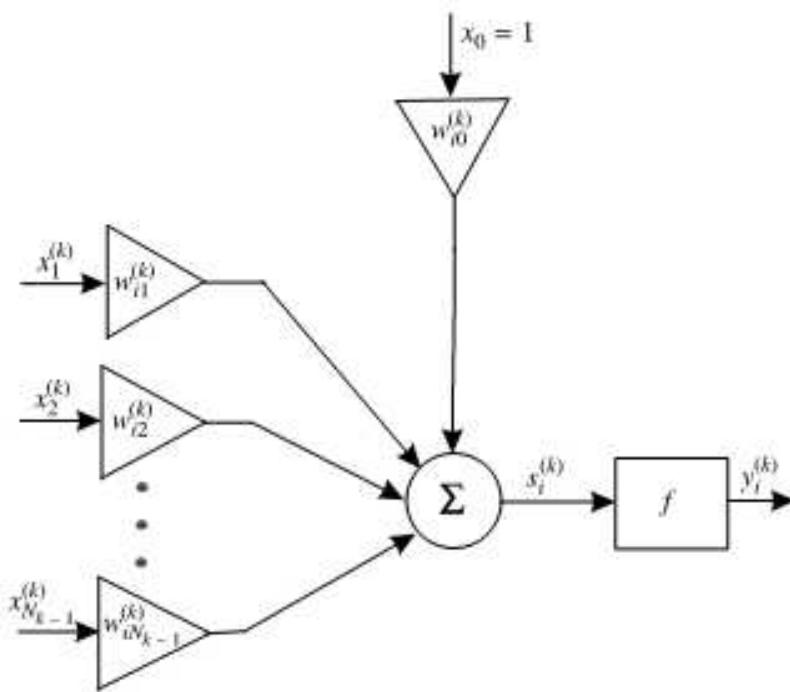
Sygnał wyjściowy i -tego neuronu w k -tej warstwie jest oznaczony jako $y_i^{(k)}(t)$, $i = 1, \dots, N_k$, $k = 1, \dots, L$.

Na rysunku 6.17 pokazano schemat szczegółowy i -tego neuronu w k -tej warstwie. Neuron N_i^k ma N_k wejść tworzących wektor

$$\mathbf{x}^{(k)}(t) = [x_0^{(k)}(t), \dots, x_{N_{k-1}}^{(k)}(t)]^T, \quad (6.103)$$

przy czym $x_i^{(k)}(t) = +1$ dla $i = 0$ oraz $k = 1, \dots, L$. Zwrócić uwagę na fakt, że sygnał wejściowy neuronu N_i^k jest powiązany z sygnałem wyjściowym warstwy $k-1$ w sposób następujący:

$$x_i^{(k)}(t) = \begin{cases} x_i(t) & \text{dla } k = 1, \\ y_i^{(k-1)}(t) & \text{dla } k = 2, \dots, L, \\ +1 & \text{dla } i = 0, k = 1, \dots, L. \end{cases} \quad (6.104)$$

Rys. 6.17. Schemat neuronu N_i^k

Na rysunku 6.17 $w_{ij}^{(k)}(t)$ jest wagą i -tego neuronu, $i = 1, \dots, N_k$, warstwy k , łączącą ten neuron z j -ym sygnałem wejściowym $x_j^{(k)}(t)$, $j = 0, 1, \dots, N_k$. Wektor wag neuronu N_i^k oznaczmy

$$\mathbf{w}_i^{(k)}(t) = [w_{i,0}^{(k)}(t), \dots, w_{i,N_{k-1}}^{(k)}(t)]^T, \quad k = 1, \dots, L, \quad i = 1, \dots, N_k. \quad (6.105)$$

Sygnał wyjściowy neuronu N_i^k w chwili t , $t = 1, 2, \dots$, jest określony jako

$$y_i^{(k)}(t) = f(s_i^{(k)}(t)), \quad (6.106)$$

przy czym

$$s_i^{(k)}(t) = \sum_{j=0}^{N_{k-1}} w_{ij}^{(k)}(t) x_j^{(k)}(t). \quad (6.107)$$

Zauważmy, że sygnały wyjściowe neuronów w warstwie L -tej

$$y_1^L(t), y_2^L(t), \dots, y_{N_L}^L(t) \quad (6.108)$$

są jednocześnie sygnałami wyjściowymi całej sieci. Są one porównywane z tzw. *sygnałami wzorcowymi sieci*

$$d_1^L(t), d_2^L(t), \dots, d_{N_L}^L(t). \quad (6.109)$$

Błąd na wyjściu sieci Q zdefiniujemy następująco:

$$Q(t) = \sum_{i=1}^{N_L} \varepsilon_i^{(L)^2}(t) = \sum_{i=1}^{N_L} (d_i^{(L)}(t) - y_i^{(L)}(t))^2. \quad (6.110)$$

Korzystając z zależności (6.101) i (6.110), otrzymujemy

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) - \eta \frac{\partial Q(t)}{\partial w_{ij}^{(k)}(t)}. \quad (6.111)$$

Zauważmy, że

$$\frac{\partial Q(t)}{\partial w_{ij}^{(k)}(t)} = \frac{\partial Q(t)}{\partial s_i^{(k)}(t)} \frac{\partial s_i^{(k)}(t)}{\partial w_{ij}^{(k)}(t)} = \frac{\partial Q(t)}{\partial s_i^{(k)}(t)} x_j^{(k)}(t). \quad (6.112)$$

Oznaczając

$$\delta_i^{(k)}(t) = -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^{(k)}(t)}, \quad (6.113)$$

otrzymujemy równość

$$\frac{\partial Q(t)}{\partial w_{ij}^{(k)}(t)} = -2\delta_i^{(k)}(t)x_j^{(k)}(t), \quad (6.114)$$

a zatem algorytm (6.111) możemy napisać następująco:

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + 2\eta \delta_i^{(k)}(t)x_j^{(k)}(t). \quad (6.115)$$

Sposób wyznaczania wartości $\delta_i^{(k)}$ jest uzależniony od warstwy sieci. Dla warstwy ostatniej otrzymujemy

$$\begin{aligned} \delta_i^{(L)}(t) &= -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^{(L)}(t)} = -\frac{1}{2} \frac{\partial \sum_{m=1}^{N_L} Q_m^{(L)}(t)}{\partial s_i^{(L)}(t)} = \\ &= -\frac{1}{2} \frac{\partial Q_i^{(L)}(t)}{\partial s_i^{(L)}(t)} = -\frac{1}{2} \frac{\partial (d_i^{(L)}(t) - y_i^{(L)}(t))^2}{\partial s_i^{(L)}(t)} = \\ &= Q_i^{(L)}(t) \frac{\partial y_i^{(L)}(t)}{\partial s_i^{(L)}(t)} = Q_i^{(L)}(t) f'(s_i^{(L)}(t)). \end{aligned} \quad (6.116)$$

Dla dowolnej warstwy $k \neq L$ mamy

$$\begin{aligned} \delta_i^{(k)}(t) &= -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^{(k)}(t)} = -\frac{1}{2} \sum_{m=1}^{N_{k+1}} \frac{\partial Q(t)}{\partial s_m^{(k+1)}(t)} \frac{\partial s_m^{(k+1)}(t)}{\partial s_i^{(k)}(t)} = \\ &= \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t) f'(s_i^{(k)}(t)) = \\ &= f'(s_i^{(k)}(t)) \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t). \end{aligned} \quad (6.117)$$

Zdefiniujemy błąd w warstwie k -tej (z wyjątkiem ostatniej) dla i -tego neuronu

$$\varepsilon_i^{(k)}(t) = \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t), \quad k = 1, \dots, L-1. \quad (6.118)$$

Podstawiając wyrażenie (6.118) do wzoru (6.117), otrzymujemy

$$\delta_i^{(k)}(t) = \varepsilon_i^{(k)}(t) f'(s_i^{(k)}(t)). \quad (6.119)$$

W rezultacie algorytm wstecznej propagacji błędów możemy zapisać następująco:

$$y_i^{(k)}(t) = f(s_i^{(k)}(t)), \quad s_i^{(k)}(t) = \sum_{j=0}^{N_{k-1}} w_{ij}^{(k)}(t) x_j^{(k)}(t), \quad (6.120)$$

$$Q_i^{(k)}(t) = \begin{cases} d_i^{(L)}(t) - y_i^{(L)}(t) & \text{dla } k = L, \\ \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t) & \text{dla } k = 1, \dots, L-1, \end{cases} \quad (6.121)$$

$$\delta_i^{(k)}(t) = \varepsilon_i^{(k)}(t) f'(s_i^{(k)}(t)), \quad (6.123)$$

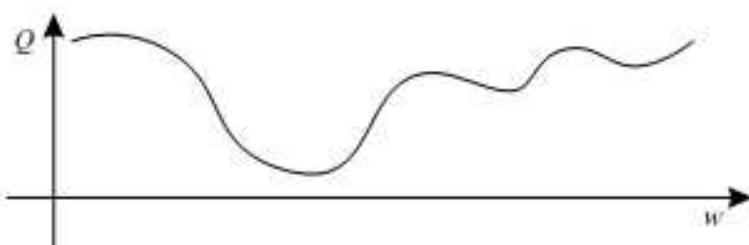
$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + 2\eta \delta_i^{(k)}(t) x_j^{(k)}(t). \quad (6.124)$$

Powyżej przedstawiliśmy szereg zależności matematycznych, opisujących sposób uczenia wielowarstwowej sieci neuronowej. Działanie algorytmu zaczyna się od podania wzorca uczącego na wejście sieci. Najpierw zostaje on przetworzony przez neurony warstwy pierwszej. Przez pojęcie przetwarzania rozumiemy tutaj oczywiście wyznaczenie sygnału wyjściowego (wzory (6.106) i (6.107)) dla każdego neuronu w danej warstwie. Otrzymane w ten sposób sygnały stają się wejściami dla neuronów warstwy następnej. Cykl ten powtarza się, tzn. ponownie wyznaczamy wartości sygnałów na wyjściach neuronów kolejnej warstwy i przekazujemy je dalej, aż do warstwy ostatniej. Znając sygnał wyjściowy warstwy ostatniej oraz sygnał wzorcowy z ciągu uczącego, możemy obliczyć błąd na wyjściu sieci zgodnie ze wzorem (6.121). Wykorzystując regułę delta, podobnie jak dla pojedynczego neuronu sigmoidalnego, możemy zmodyfikować wagę neuronów ostatniej warstwy, korzystając ze wzorów (6.121), (6.123), (6.124). Jednak w ten sposób nie zmodyfikujemy wag w neuronach warstw ukrytych (nie znamy wartości $\delta_i^{(k)}$ dla neuronów tych warstw), a przecież funkcja celu określona wzorem (6.110) jest funkcją, w której zmiennymi są wszystkie wagi sieci. Dlatego błąd wyjściowy jest propagowany od tyłu (od warstwy wyjściowej do wejściowej) zgodnie z połączeniami neuronów między warstwami i z uwzględnieniem ich funkcji aktywacji (patrz wzory: (6.122), (6.123), (6.124)). Nazwa algorytmu pochodzi od sposobu jego realizacji, tzn. błąd jest „cofany” od warstwy wyjściowej do wejściowej.

W dyskusji dotyczącej metody wstecznej propagacji błędów stwierdziliśmy, iż uczenie sieci (modyfikację wag) przeprowadza się każdorazowo po podaniu na wejście wektora uczącego. Postępowanie takie nosi nazwę *przyrostowego uaktualniania wag*. Istnieje jednak inny sposób postępowania. Mianowicie na wejście sieci można kolejno podawać wektory uczące, wyznaczać odpowiadające im sygnały na wyjściu sieci, a następnie po porównaniu ich z wartościami wzorcowymi sumować otrzymywane w kolejnych iteracjach błędy. Gdy skończymy już podawanie na wejście sieci próbek z danej epoki, dokonujemy korekty wartości wszystkich wag z wykorzystaniem skumulowanej wartości błędu. Ten algorytm nosi nazwę *kumulacyjnego uaktualniania wag*.

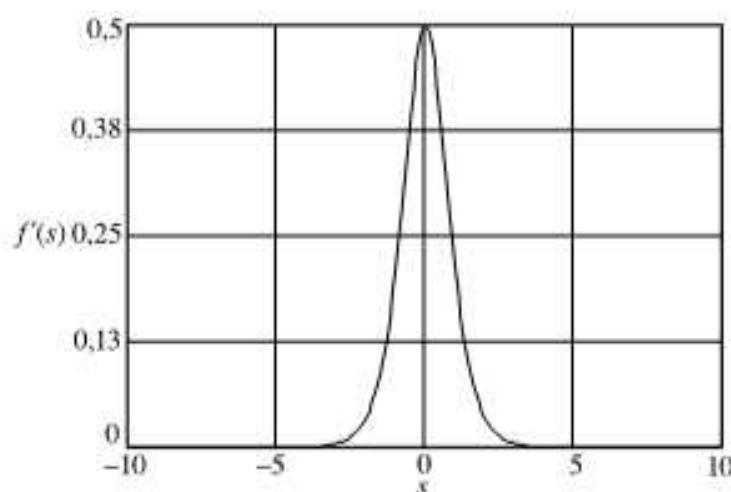
Bardzo ważnym problemem, często poruszany w artykułach dotyczących sieci neuronowych, jest inicjalizacja wag początkowych sieci. Oczywistym faktem jest, że im wartości startowe wag będą bliższe wartości optymalnych (minimalizujących funkcję błędu (6.110)), tym uczenie będzie trwało krócej. Jak wspominaliśmy wcześniej, uczenie

sieci polega na znalezieniu minimum funkcji błędu. Jest to funkcja wielu zmiennych, którymi są wszystkie wagi. Funkcja taka może mieć wiele minimów lokalnych. Na rysunku 6.18 przedstawiono hipotetyczny wykres funkcji błędu w przypadku jednej zmiennej (jednej wagi).



Rys. 6.18. Hipotetyczna funkcja błędu jednej zmiennej

Nawet w tak prostym przypadku może istnieć wiele minimów lokalnych, co zdarza się bardzo często. Niestety, metoda największego spadku nie jest metodą odporną na występowanie minimów lokalnych i algorytm, poszukując rozwiązania, może w nich utknąć. Dlatego niejednokrotnie stosuje się modyfikacje algorytmu wstecznej propagacji błędów, które zostaną omówione w dalszej części rozdziału. Najprostszą metodą, która może wyeliminować problem minimów lokalnych, jest rozpoczęwanie uczenia sieci neuronowej dla różnych wartości wag początkowych. Postępujemy więc w następujący sposób. Wybieramy wagi losowo, zakładając rozkład równomierny w określonym przedziale liczbowym. Następnie, używając algorytmu wstecznej propagacji błędów, uczymy sieć neuronową. Gdy błąd uczenia przestaje maleć lub zaczyna rosnąć, ponownie wybieramy losowo wagi, tylko tym razem z innego przedziału ich wartości. Powtarzamy uczenie sieci i obserwujemy, czy końcowa wartość błędu w tym cyklu jest mniejsza od poprzedniej. Jeżeli tak, to za pierwszym razem mogliśmy utknąć w minimum lokalnym. Oczywiście opisany tok postępowania wydłuża proces uczenia sieci. Wybór dużych wartości wag początkowych powoduje, że błąd średni kwadratowy na wyjściu sieci jest praktycznie stały. Może to być spowodowane tym, że sygnał wyjściowy części liniowej neuronu jest bardzo duży, a więc mamy do czynienia z nasyceniem funkcji aktywacji. Zmiana wartości wag w algorytmie wstecznej propagacji błędów jest proporcjonalna do pochodnej tejże funkcji. Jak pokazano na rysunku 6.19, pochodna ta ma kształt krzywej dzwonowej ze środkiem w punkcie zero.



Rys. 6.19. Wykres pochodnej sigmoidalnej funkcji aktywacji

Zatem, im większa jest wartość bezwzględna sygnału na wyjściu części liniowej neuronu, tym mniejsza będzie korekcja wag, a zatem uczenie będzie przebiegało bardzo wolno. Ogólnie powinno się dążyć, aby wagi początkowe wybrane losowo dawały sygnał bliski jedności na wyjściu części liniowej neuronu. Elementem, który ma bardzo duży wpływ na zbieżność algorytmu wstecznej propagacji błędów, jest współczynnik uczenia η . Niestety, nie istnieje ogólna metoda doboru jego wartości. Z reguły krok korekcji przyjmuje się z przedziału $(0, 1)$. Jeżeli mamy do czynienia z płaską funkcją celu, to wartości gradientu są niewielkie i przy większych wartościach współczynnika uczenia algorytm szybciej znajdzie rozwiązanie. Natomiast, jeżeli funkcja celu jest stroma, to przyjęcie dużej wartości współczynnika spowoduje oscylacje wokół rozwiązania, a tym samym przedłużenie procesu uczenia. Współczynnik uczenia dobiera się w zależności od problemu, który należy rozwiązać, i w dużym stopniu zależy od doświadczenia uczącego sieć.

Ważnym zagadnieniem jest dobór kryterium stopu algorytmu wstecznej propagacji błędów. Jest oczywiste, że w przypadku osiągnięcia minimum (lokalnego lub globalnego) wektor gradientu (6.96) przyjmuje wartość **0**. Zatem działanie algorytmu możemy zatrzymać, jeżeli wartość normy euklidesowej wektora gradientu spadnie poniżej zadanego progu. Alternatywnie możemy sprawdzać, czy błąd średni kwadratowy wyznaczony w ramach jednej epoki spadł poniżej innego zadanego progu. Niekiedy stosuje się kombinację obu wymienionych metod, tzn. algorytm zostaje zatrzymany, jeżeli jedna z wymienionych wartości spadnie poniżej założonego progu. Inna propozycja polega na ciągłym testowaniu sieci neuronowej w trakcie uczenia (po każdej iteracji). Proces uczenia można przerwać, gdy sieć wykazuje dobre właściwości generalizacji.

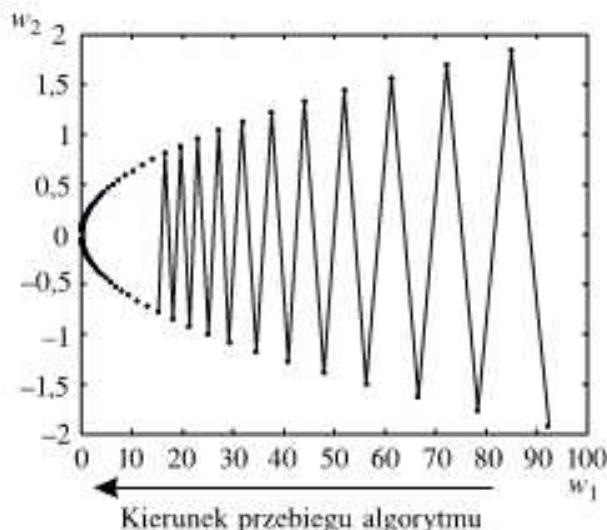
6.3.3. Algorytm wstecznej propagacji błędów z członem momentum

Jak wspomnieliśmy wcześniej, podczas uczenia sieci neuronowej algorytm wstecznej propagacji błędów często nie spisuje się najlepiej: może utknąć w minimum lokalnym lub też oscylować wokół ostatecznego rozwiązania. Dlatego też do standardowej postaci tej metody wprowadza się dodatkowy współczynnik α zwany *momentum*, zgodnie z którym wzór (6.124) modyfikujemy następująco:

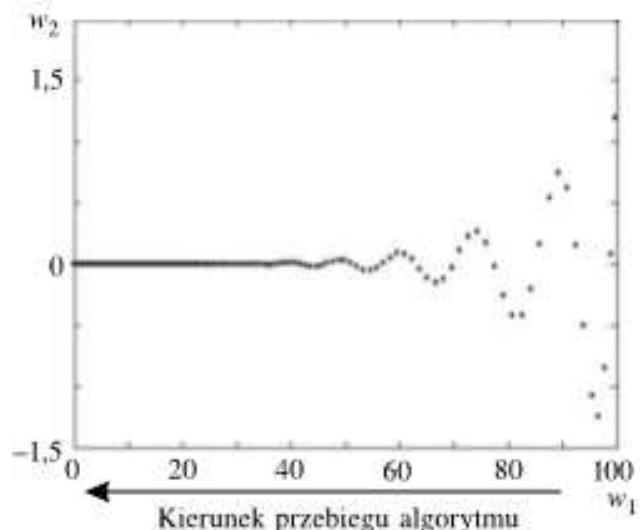
$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + 2\eta\delta_i^{(k)}x_j^{(k)}(t) + \alpha[w_{ij}^{(k)}(t) - w_{ij}^{(k)}(t-1)]. \quad (6.125)$$

Współczynnik ten uzależnia wartość wagi w kroku następnym ($t+1$) nie tylko od jej wartości w kroku obecnym (jak w klasycznej metodzie wstecznej propagacji błędów), ale też w kroku poprzednim ($t-1$). Jeżeli w kolejnych iteracjach kierunek modyfikacji wagi był ten sam, to składnik zawierający współczynnik momentum powoduje wzrost wartości przyrostu wagi i przesunięcie jej z większą siłą w kierunku minimum. Natomiast w przypadku odwrotnym składnik ten powoduje zahamowanie gwałtownych zmian wag. Ogólnie działanie członu momentum uaktywnia się na płaskich odcinkach funkcji celu, jak również w pobliżu minimum lokalnego. W szczególności na płaskich odcinkach funkcji celu dzięki temu członowi następuje znaczne przyspieszenie procesu uczenia. Natomiast w pobliżu minimum lokalnego wartość gradientu funkcji celu jest bliska zera i człon momentum staje się dominujący we wzorze (6.125), co pozwala na

opuszczenie obszaru minimum lokalnego. Współczynnik α przyjmuje wartości z zakresu $(0, 1)$, najczęściej zakłada się $\alpha = 0,9$. Ustalenie jego konkretnej wartości zależy od rozważanego problemu oraz doświadczenia osoby przeprowadzającej uczenie.



Rys. 6.20. Ilustracja do przykładu 6.3 — algorytm największego spadku



Rys. 6.21. Ilustracja do przykładu 6.3 — algorytm momentum

Przykład 6.3

Na rysunku 6.20 przedstawiamy przebieg algorytmu poszukiwania minimum funkcji $F(w_1, w_2) = w_1^2 + w_2^{0,08}$ metodą największego spadku. Współczynnik uczenia η jest równy 0,9, a punkt startowy $[w_1(0), w_2(0)] = [-2, 90]$. Przy takim doborze współczynnika η kolejne rozwiązania wolno zmierzają w kierunku punktu optymalnego $\mathbf{w} = [0, 0]^T$, a cały przebieg charakteryzuje się występowaniem oscylacji. Rysunek 6.21 obrazuje rozwiązanie tego samego problemu z wykorzystaniem algorytmu (6.125), w którym $\eta = 0,2$, a $\alpha = 0,9$. Jak można zauważyć na tym rysunku, współczynnik momentum powoduje, iż algorytm jest szybciej zbieżny, a także nie występują oscylacje wokół minimum.

6.3.4. Algorytm zmiennej metryki

Kolejną modyfikacją algorytmu wstecznej propagacji błędów jest algorytm zmiennej metryki. W celu wyprowadzenia tego algorytmu rozważmy trzy pierwsze składniki w rozwinięciu funkcji celu w szereg Taylora. Funkcja celu będzie teraz zdefiniowana następująco:

$$Q(\mathbf{w}(t) + \mathbf{p}(t)) = Q(\mathbf{w}(t)) + (\mathbf{g}(\mathbf{w}(t)))^T \mathbf{p}(t) + \frac{1}{2} \mathbf{p}^T \mathbf{H}(\mathbf{w}(t)) \mathbf{p}(t). \quad (6.126)$$

Przypomnijmy sobie teraz warunki konieczne i wystarczające istnienia minimum funkcji. Warunkiem koniecznym jest zerowanie się pierwszej pochodnej w tym punkcie. Niestety, nie jest to warunek dostateczny, co łatwo sprawdzić na przykładzie funkcji $f(x) = x^3$ (w punkcie $x = 0$ pierwsza pochodna jest równa zero, ale punkt ten nie jest minimum). Warunkiem dostatecznym istnienia minimum funkcji w punkcie x_0 jest

spełnienie dwóch warunków: 1) zerowanie się pierwszej pochodnej w tym punkcie oraz 2) wartość drugiej pochodnej w tym punkcie powinna być większa od 0. Analogiczne warunki muszą być spełnione w przypadku funkcji wielu zmiennych. Wyprowadzimy teraz algorytm zmiennej metryki. Zminimalizujemy kryterium Q dane wzorem (6.126) względem wektora \mathbf{p} . Oczywiście

$$\frac{\partial Q(\mathbf{w}(t) + \mathbf{p}(t))}{\partial \mathbf{p}(t)} = (\mathbf{g}(\mathbf{w}(t)))^T + \mathbf{H}(\mathbf{w}(t))\mathbf{p}(t). \quad (6.127)$$

Wektor $\mathbf{p}(t)$ minimalizujący kryterium Q musi spełniać równanie

$$\mathbf{g}(\mathbf{w}(t)) + \mathbf{H}(\mathbf{w}(t))\mathbf{p}(t) = \mathbf{0}. \quad (6.128)$$

W konsekwencji

$$\mathbf{p}(t) = -[\mathbf{H}(\mathbf{w}(t))]^{-1}\mathbf{g}(\mathbf{w}(t)). \quad (6.129)$$

Aby funkcja celu $Q(\mathbf{w}(t) + \mathbf{p}(t))$ osiągnęła minimum w danym punkcie, hesjan (macierz drugich pochodnych) powinien być w tym punkcie dodatnio określony. Niestety, ten warunek jest bardzo trudno spełnić. Dlatego w praktyce wyznacza się przybliżoną jego wartość $\mathbf{G}(\mathbf{w}(t))$. Przyjmijmy, że $\mathbf{c}(t) = \mathbf{w}(t) - \mathbf{w}(t-1)$, $\mathbf{r}(t) = \mathbf{g}(\mathbf{w}(t)) - \mathbf{g}(\mathbf{w}(t-1))$, $\mathbf{V}(t) = [\mathbf{G}(\mathbf{w}(t))]^{-1}$ oraz $\mathbf{V}(t-1) = [\mathbf{G}(\mathbf{w}(t-1))]^{-1}$. Do aktualniania wartości macierzy \mathbf{V} stosuje się dwie znane metody:

i) metodę Davidona–Fletchera–Powella

$$\mathbf{V}(t) = \mathbf{V}(t-1) + \frac{\mathbf{c}(t)\mathbf{c}^T(t)}{\mathbf{c}^T(t)\mathbf{r}(t)} - \frac{\mathbf{V}(t-1)\mathbf{r}(t)\mathbf{r}^T(t)\mathbf{V}(t-1)}{\mathbf{r}^T(t)\mathbf{V}(t-1)\mathbf{r}(t)}; \quad (6.130)$$

ii) metodę Broydena–Fletcher–Goldfarba–Shanno

$$\begin{aligned} \mathbf{V}(t) = \mathbf{V}(t-1) + & \left[1 + \frac{\mathbf{r}^T(t)\mathbf{V}(t-1)\mathbf{r}(t)}{\mathbf{c}^T(t)\mathbf{r}(t)} \right] \frac{\mathbf{c}(t)\mathbf{c}^T(t)}{\mathbf{c}^T(t)\mathbf{r}(t)} \\ & - \frac{\mathbf{c}(t)\mathbf{r}^T(t)\mathbf{V}(t-1) + \mathbf{V}(t-1)\mathbf{r}(t)\mathbf{c}^T(t)}{\mathbf{c}^T(t)\mathbf{r}(t)}, \end{aligned} \quad (6.131)$$

przy czym $\mathbf{V}(0) = \mathbf{I}$. Łącząc wzór (6.129) ze wzorem (6.130) lub (6.131), otrzymujemy

$$\mathbf{p}(t) = -\mathbf{V}(t)\mathbf{g}(\mathbf{w}(t)). \quad (6.132)$$

Algorytm zmiennej metryki charakteryzuje się dosyć szybką zbieżnością. Jego wadą jest stosunkowo duża złożoność obliczeniowa, związana z koniecznością wyznaczania w każdym kroku wszystkich elementów hesjanu. Dlatego algorytm ten stosuje się do niezbyt dużych sieci.

6.3.5. Algorytm Levenberga–Marquardta

Podobnie jak w poprzednich punktach przyjmiemy, że $Q(\mathbf{w})$ jest funkcją błędu, i będziemy dążyć do znalezienia jej minimum. W algorytmie Levenberga–Marquardta wykorzystuje się rozwinięcie funkcji $Q(\mathbf{w})$, wyrażonej wzorem (6.95), do trzeciego składnika. Poszukiwanie minimum w ten sposób otrzymanego wyrażenia opisaliśmy już w po-

przednim punkcie. Kierunek minimalizacji jest identyczny jak w przypadku algorytmu zmiennej metryki, tzn. $\mathbf{p}(t) = -[\mathbf{H}(\mathbf{w}(t))]^{-1}\mathbf{g}(\mathbf{w}(t))$.

Rozważmy sieć neuronową mającą N_L wyjść. Przyjmujemy miarę błędu w postaci

$$Q(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N_L} e_i^2(\mathbf{w}), \quad (6.133)$$

gdzie

$$e_i = d_i - y_i. \quad (6.134)$$

W literaturze podaje się następujące wzory określające wektor gradientu:

$$\mathbf{g}(\mathbf{w}) = [J(\mathbf{w})]^T e(\mathbf{w}) \quad (6.135)$$

oraz aproksymowaną macierz hesjanu

$$\mathbf{H}(\mathbf{w}) = [J(\mathbf{w})]^T J(\mathbf{w}) + S(\mathbf{w}), \quad (6.136)$$

gdzie

$$J(\mathbf{w}) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_n} \\ \frac{\partial e_3}{\partial w_1} & \frac{\partial e_3}{\partial w_2} & \cdots & \frac{\partial e_3}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{N_L}}{\partial w_1} & \frac{\partial e_{N_L}}{\partial w_2} & \cdots & \frac{\partial e_{N_L}}{\partial w_n} \end{bmatrix} \quad (6.137)$$

oraz

$$e(\mathbf{w}) = [e_1(\mathbf{w}), \dots, e_{N_L}(\mathbf{w})]^T. \quad (6.138)$$

Człon $S(\mathbf{w})$ we wzorze (6.136) odpowiada składnikom rozwinięcia hesjanu, zawierającym wyższe pochodne względem wektora \mathbf{w} . Człon ten można aproksymować jako $S(\mathbf{w}) = \mu \mathbf{I}$, gdzie μ jest tzw. parametrem Levenberga–Marquardta. Korzystając z tej aproksymacji oraz podstawiając zależności (6.135) i (6.136) do wzoru (6.129), otrzymujemy

$$\mathbf{p}(t) = -[J^T(\mathbf{w}(t)) J(\mathbf{w}(t)) + \mu(t) \mathbf{I}]^{-1} [J^T(\mathbf{w}(t))]^T e(\mathbf{w}(t)). \quad (6.139)$$

Parametr μ dobiera się w zależności od błędu na wyjściu sieci podczas procesu jej uczenia. Parametr ten przyjmuje duże wartości na starcie algorytmu, w miarę zbliżania się do rozwiązania optymalnego jego wartość maleje aż do wartości zerowej.

Przykład 6.4

Dotychczas rozważane algorytmy porównamy na przykładzie problemu XOR. Doświadczenia powtarzaliśmy dziesięciokrotnie dla każdego algorytmu (zmieniałyśmy parametry uczenia, np. współczynnik uczenia) i wybraliśmy najlepsze wyniki. Wszystkie doświadczenia wykonywaliśmy z wykorzystaniem pakietu Matlab. Uczenie przerywaliśmy, kiedy błąd średni kwadratowy spadał poniżej 0,012. Rezultaty obrazuje tabela 6.3. Jak łatwo zauważać, najszybszą metodą jest metoda Levenberga–Marquardta. Niestety, nie zawsze ten algorytm możemy stosować, gdyż wymaga on dużych rozmiarów pamięci operacyjnej komputera.

Tabela 6.3. Porównanie działania algorytmów uczenia sieci neuronowych

Nazwa algorytmu uczącego	Liczba epok
Największego spadku	415
Momentum	250
Zmiennej metryki	8
Levenberga–Marquardta	3

6.3.6. Rekurencyjna metoda najmniejszych kwadratów

W punktach 6.2.3 i 6.2.4 przedstawiliśmy modele neuronów uczonych metodą RLS. Obecnie metodę tę zastosujemy do uczenia wielowarstwowych sieci neuronowych. Jako miarę błędu przyjmujemy

$$Q(t) = \sum_{l=1}^t \lambda^{t-l} \sum_{j=1}^{N_L} \varepsilon_j^{(L)^2}(l) = \sum_{l=1}^t \lambda^{t-l} \sum_{j=1}^{N_L} [d_j^{(L)}(l) - f(\mathbf{x}^{(L)^T}(l) \mathbf{w}_j^{(L)}(t))]^2, \quad (6.140)$$

gdzie λ jest współczynnikiem zapominania wybieranym z przedziału $(0, 1]$. Zauważmy, że poprzednie błędy mają mniejszy wpływ na wartość wyrażenia (6.140). W toku dalszych rozważań będziemy stosować oznaczenia wprowadzone w punkcie 6.3.2. Ponadto oznaczmy

$$\varepsilon_i^{(k)}(l) = d_i^{(k)}(l) - y_i^{(k)}(l) \quad (6.141)$$

oraz

$$b_i^{(k)}(l) = f^{-1}(d_i^{(k)}(l)), \quad (6.142)$$

zakładając, że f jest funkcją odwracalną, $l = 1, \dots, t$, $i = 1, \dots, N_k$, $k = 1, \dots, L$.

Obliczając gradient miary błędu i przyrównując go do zera, otrzymujemy równanie

$$\frac{\partial Q(t)}{\partial \mathbf{w}_i^{(k)}(t)} = 2 \sum_{l=1}^t \lambda^{t-l} \sum_{j=1}^{N_L} \frac{\partial \varepsilon_j^{(L)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_j^{(L)}(l) = -2 \sum_{l=1}^t \lambda^{t-l} \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_j^{(L)}(l) = \mathbf{0}. \quad (6.143)$$

Korzystając z zależności (6.106) i (6.107), równanie (6.143) przekształcamy następująco:

$$\begin{aligned} \sum_{l=1}^t \lambda^{t-l} \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(l)}{\partial s_j^{(L)}(l)} \sum_{p=1}^{N_{L-1}} \frac{\partial s_p^{(L)}(l)}{\partial y_p^{(L-1)}(l)} \frac{\partial y_p^{(L-1)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_j^{(L)}(l) \\ = \sum_{l=1}^t \lambda^{t-l} \sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \sum_{j=1}^{N_L} \frac{\partial y_j^{(L-1)}(l)}{\partial s_j^{(L)}(l)} w_{jp}^{(L)} \varepsilon_j^{(L)}(l) \end{aligned} \quad (6.144)$$

$$= \sum_{l=1}^t \lambda^{t-l} \sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_p^{(L-1)}(l) = \sum_{l=1}^t \lambda^{t-l} \sum_{q=1}^{N_k} \frac{\partial y_q^{(k)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_q^{(k)}(l) = \mathbf{0},$$

gdzie

$$\varepsilon_p^{(k)}(l) = \sum_{j=1}^{N_{k+1}} \frac{\partial y_j^{(k+1)}(l)}{\partial s_j^{(k+1)}(l)} w_{jp}^{(k+1)}(t) \varepsilon_j^{(k+1)}(l). \quad (6.145)$$

Wyrażenie (6.145) określa sposób wyznaczania błędów w kolejnych warstwach, zaczynając od ostatniej. Przekształcając dalej, otrzymujemy ciąg równości

$$\begin{aligned} \sum_{l=1}^t \lambda^{t-l} \sum_{q=1}^{N_k} \frac{\partial y_q^{(k)}(l)}{\partial \mathbf{w}_i^{(k)}(t)} \varepsilon_q^{(k)}(l) &= \sum_{j=1}^t \lambda^{t-l} \frac{\partial y_i^{(k)}(l)}{\partial s_i^{(k)}(t)} \mathbf{y}^{(k-1)^T}(l) \varepsilon_i^{(k)}(l) \\ &= \sum_{l=1}^t \lambda^{t-l} \frac{\partial y_i^{(k)}(l)}{\partial s_i^{(k)}(t)} \mathbf{y}^{(k-1)^T}(l) [d_i^{(k)}(l) - y_i^{(k)}(l)] = \mathbf{0}, \end{aligned} \quad (6.146)$$

w których

$$\mathbf{y}^{(k)} = [y_1^{(k)}, \dots, y_{N_k}^{(k)}]^T.$$

Stosując aproksymację

$$f(b_i^{(k)}(l)) \approx f(s_i^{(k)}(l)) + f'(s_i^{(k)}(l))(b_i^{(k)}(l) - s_i^{(k)}(l)), \quad (6.147)$$

otrzymujemy równanie normalne

$$\sum_{l=1}^t \lambda^{t-l} f'^2(s_i^{(k)}(l)) [b_i^{(k)}(l) - \mathbf{x}^{(k)^T}(l) \mathbf{w}_i^{(k)}(t)] \mathbf{x}^{(k)^T}(l) = \mathbf{0}. \quad (6.148)$$

którego wektorowy zapis jest postaci

$$\mathbf{r}_i^{(k)}(t) = \mathbf{R}_i^{(k)}(t) \mathbf{w}_i^{(k)}(t), \quad (6.149)$$

gdzie

$$\mathbf{R}_i^{(k)}(t) = \sum_{l=1}^t \lambda^{t-l} f'^2(s_i^{(k)}(l)) \mathbf{x}^{(k)}(l) \mathbf{x}^{(k)^T}(l), \quad (6.150)$$

$$\mathbf{r}_i^{(k)}(t) = \sum_{l=1}^t \lambda^{t-l} f'^2(s_i^{(k)}(l)) b_i^{(k)}(l) \mathbf{x}^{(k)}(l). \quad (6.151)$$

Równanie (6.149) możemy rozwiązać w sposób rekurencyjny, bez konieczności odwracania macierzy $\mathbf{R}_i^{(k)}(t)$. Wymaga to zastosowania algorytmu RLS, analogicznie jak to zrobiono w punktach 6.2.3 i 6.2.4 w przypadku modelu pojedynczego neuronu. W re-

zultacie adaptacyjna korekcja wszystkich wag $\mathbf{w}_i^{(k)}$ jest przeprowadzona następująco:

$$\varepsilon_i^{(k)}(t) = \begin{cases} d_i^{(L)}(t) - y_i^{(L)}(t) & \text{dla } k = L, \\ \sum_{j=1}^{N_{k+1}} f'(s_j^{(k+1)}(t)) w_{ji}^{(k+1)}(t) \varepsilon_j^{(k+1)}(t) & \text{dla } k = 1, \dots, L-1, \end{cases} \quad (6.152)$$

$$\mathbf{g}_i^{(k)}(t) = \frac{f'(s_i^{(k)}(t)) \mathbf{P}_i^{(k)}(t-1) \mathbf{x}^{(k)}(t)}{\lambda + f'^2(s_i^{(k)}(t)) \mathbf{x}^{(k)T}(t) \mathbf{P}_i^{(k)}(t-1) \mathbf{x}^{(k)}(t)}, \quad (6.153)$$

$$\mathbf{P}_i^{(k)}(t) = \lambda^{-1} [\mathbf{I} - f'(s_i^{(k)}(t)) \mathbf{g}_i^{(k)}(t) \mathbf{x}^{(k)T}(t)] \mathbf{P}_i^{(k)}(t-1), \quad (6.154)$$

$$\mathbf{w}_i^{(k)}(t) = \mathbf{w}_i^{(k)}(t-1) + \mathbf{g}_i^{(k)}(t) \varepsilon_i^{(k)}(t) \text{ gdzie } i = 1, \dots, N_k, k = 1, \dots, L. \quad (6.155)$$

Wartości początkowe algorytmu RLS przyjmuje się zwykle następująco:

$$\mathbf{P}^{(k)}(0) = \delta \mathbf{I}, \quad \delta \gg 0, \quad (6.156)$$

$$\mathbf{w}_i^{(k)}(0) = 0. \quad (6.157)$$

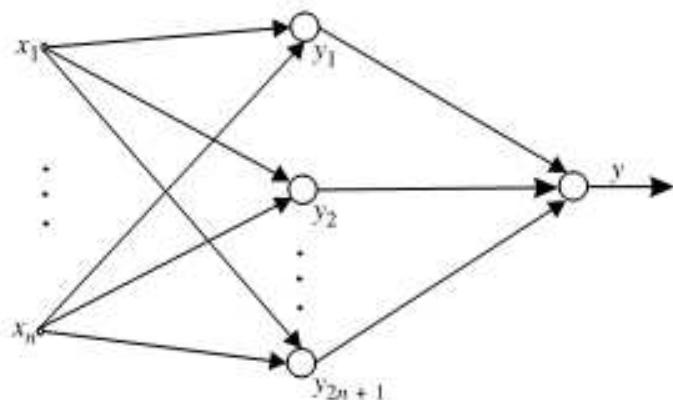
Wagi początkowe $\mathbf{w}_i^{(k)}(0)$ sieci neuronowej mogą być również wybrane losowo z pewnego przedziału.

6.3.7. Dobór architektury sieci

Przez pojęcie projektowania architektury sieci neuronowych czytelnik powinien rozumieć dobór liczby warstw sieci oraz dobór liczby neuronów w każdej warstwie. Jak można przypuszczać, wielkości te są uzależnione od problemu, który staramy się rozwiązać. Zastanówmy się, jak wpływa liczba warstw sieci na sposób jej działania. Zgodnie z tym, co napisaliśmy wcześniej, pojedynczy neuron dzieli płaszczyznę na dwie części. Natomiast dwie warstwy mogą odwzorowywać już simpleksy, czyli wypukłe ograniczone hiperplaszczyzny obszary. Za pomocą trzech warstw możemy zdefiniować dowolny obszar. Zatem trójwarstwowa sieć jest w stanie rozwiązać szeroką gamę problemów klasyfikacji oraz aproksymacji. W tym miejscu warto przytoczyć twierdzenie Kołmogorowa, zgodnie z którym dowolną ciągłą funkcję rzeczywistą $f(x_1, \dots, x_n)$, zdefiniowaną na $[0, 1]^n$, $n \geq 2$, można aproksymować za pomocą funkcji F danej wzorem

$$F(\mathbf{x}) = \sum_{j=1}^{2n+1} g_j \left(\sum_{i=1}^n \phi_{ij}(x_i) \right), \quad (6.158)$$

w którym $\mathbf{x} = [x_1, \dots, x_n]^T$, g_j , $j = 1, \dots, 2n+1$ są odpowiednio dobranymi ciągłymi funkcjami jednej zmiennej, natomiast ϕ_{ij} , $i = 1, \dots, n$, $j = 1, \dots, 2n+1$, są funkcjami ciągłymi i monotonicznie rosnącymi oraz niezależnymi od funkcji f . Przyjrzyjmy się przez moment zależności (6.158) i zastanówmy się, jak można odnieść ją do struktury sieci neuronowej. Łatwo zauważyc, że strukturę odpowiadającą zależności (6.158) tworzy sieć neuronowa dwuwarstwowa o n wejściach, $2n+1$ neuronach w warstwie ukrytej i jednym neuronie z liniową funkcją aktywacji w warstwie wyjściowej. Na podstawie powyższych rozważań możemy stwierdzić, iż struktura sieci przedstawiona na rysunku 6.22 jest w stanie aproksymować dowolną funkcję ciągłą n zmiennych zdefiniowaną na



Rys. 6.22. Struktura sieci neuronowej umożliwiająca aproksymację dowolnej funkcji ciągłej

$[0, 1]^n$. Twierdzenie Kołmogorowa ma charakter teoretyczny, gdyż nie podaje rodzaju funkcji nieliniowych oraz metod uczenia sieci.

W literaturze znane jest również inne twierdzenie, które bezpośrednio odnosi się do wielowarstwowej sieci neuronowej.

Załóżmy, że ϕ jest dowolną ciągłą funkcją sigmoidalną. Wtedy dla każdej ciągłej funkcji f zdefiniowanej na $[0, 1]^n$, $n \geq 2$, i dla dowolnego $\varepsilon > 0$, istnieje liczba całkowita N i zbiór stałych α_i , θ_i oraz w_{ij} , $i = 1, \dots, N$, $j = 1, \dots, n$, takich, że funkcja

$$F(x_1, \dots, x_n) = \sum_{i=1}^N \alpha_i \phi \left(\sum_{j=1}^n w_{ij} x_j - \theta_i \right) \quad (6.159)$$

aproksymuje funkcję f , tzn.

$$|F(x_1, \dots, x_n) - f(x_1, \dots, x_n)| < \varepsilon$$

dla wszystkich $\{x_1, \dots, x_n\} \in [0, 1]^n$. Na podstawie powyższego twierdzenia śmiało możemy powiedzieć, że sieć neuronowa z liniowym neuronem wyjściowym oraz jedną warstwą ukrytą o neuronach z sigmoidalną funkcją aktywacji może aproksymować dowolną rzeczywistą funkcję ciągłą zdefiniowaną na $[0, 1]^n$. W praktyce okazuje się, że istnieje niewiele problemów, do których rozwiązania potrzeba więcej niż dwóch warstw ukrytych.

Bardzo duży wpływ na działanie sieci ma liczba neuronów w poszczególnych warstwach. Warto zaznaczyć, że zbyt duża liczba neuronów wydłuża proces uczenia. Ponadto, jeżeli liczba próbek uczących jest niewielka w porównaniu z rozmiarem sieci, możemy strukturę „przeuczyć”, a wówczas straci ona możliwość uogólniania wiedzy. Sieć nauczy się „na pamięć” ciągu uczącego i będzie poprawnie odwzorowywać tylko próbki w nim zawarte.

Przykład 6.5

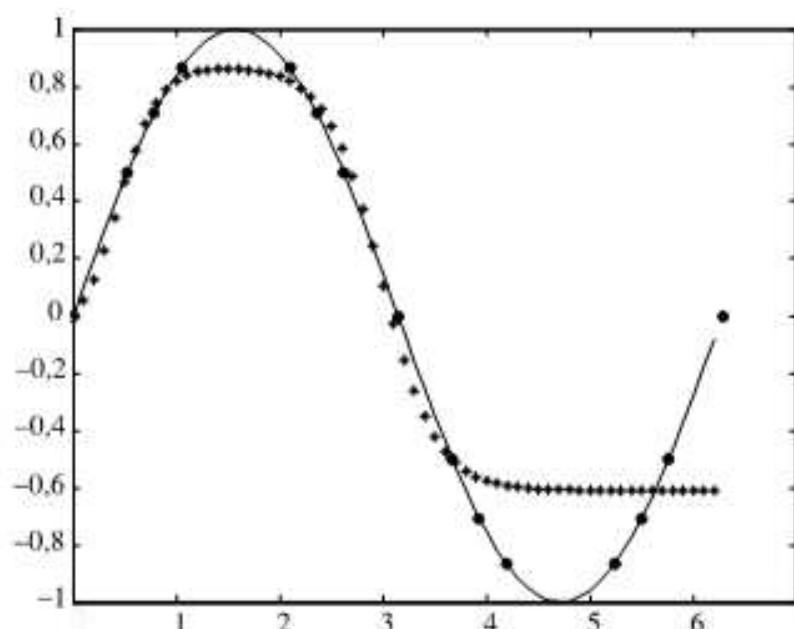
Aby przybliżyć ten problem czytelnikowi, postaramy się wykonać doświadczenie, w którym nauczymy sieć neuronową tak, aby aproksymowała funkcję $f(x) = \sin(x)$ w przedziale domkniętym $[0, 2\pi]$. Zanim zajmiemy się budowaniem odpowiedniej struktury sieci neuronowej, musimy stworzyć ciąg uczący. Jak łatwo się domyśleć, różne argumenty x funkcji $f(x) = \sin(x)$ stanowią wejścia sieci, natomiast odpowiadające im wartości wyjściowe y utożsamiamy z wartościami wzorcowymi. Wynika stąd, że ciąg uczący będzie składał się z par $\{x, \sin(x)\}$. Zauważmy, że aproksymowana przez nas funkcja jest ciągła. Nie możemy zatem ciągu uczącego zbudować ze wszystkich x z przedziału $[0, 2\pi]$ i odpowiadających im wartości y . Musimy wybrać pewną liczbę charakterystycznych par $\{x, f(x)\}$ i tymi parami uczyć sieć neuronową. W tym mo-

mencie czytelnik powinien zauważyc bardzo ważny fakt: sieć będziemy uczyć pewnym podzbiorem próbek z przedziału $[0, 2\pi]$, ale zależy nam, aby sieć poprawnie działała w całym przedziale $[0, 2\pi]$. Na tym właśnie polega zjawisko generalizacji (uogólniania) wiedzy zawartej w wagach sieci. Sieć mimo to, że uczy się tylko wybranych przykładów, potrafi swoją wiedzę uogólniać i odpowiadać poprawnie na podane na jej wejście sygnały, które nie znajdowały się w ciągu uczącym. Dlatego po nauczeniu sieci powinniśmy sprawdzić jej działanie na ciągu testowym, składającym się z próbek, które nie brały udziału w procesie uczenia. Po pozytywnym przejściu tego testu możemy stwierdzić, że struktura jest nauczona i działa poprawnie. Przyjmijmy, że ciąg uczący składa się z próbek przedstawionych w tabeli 6.4.

Tabela 6.4. Ciąg uczący w przykładzie 6.5

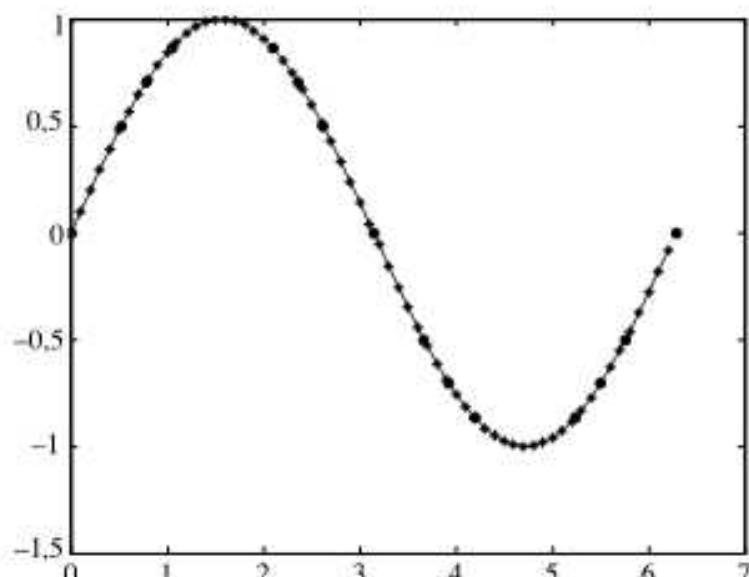
Nr próbki	1	2	3	4	5	6	7	8	9	10	11	12
Wejście x	0	$\frac{\pi}{6}$	$\frac{\pi}{3}$	$\frac{\pi}{4}$	π	2π	$\frac{7\pi}{6}$	$\frac{4\pi}{3}$	$\frac{5\pi}{4}$	$\frac{5\pi}{6}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$
Oczekiwane wyjście $d = f(x)$	0	0,5	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	0	0	-0,5	$-\frac{\sqrt{3}}{2}$	$-\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$
Nr próbki	13	14	15									
Wejście x	$\frac{5\pi}{3}$	$\frac{11\pi}{6}$	$\frac{7\pi}{4}$									
Oczekiwane wyjście $d = f(x)$	$-\frac{\sqrt{3}}{2}$	$-\frac{1}{2}$	$-\frac{\sqrt{2}}{2}$									

Struktura sieci użyta w symulacji składa się z jednej warstwy ukrytej o neuronach sigmoidalnych oraz jednego liniowego neuronu w warstwie wyjściowej. Symulacje wykonamy trzykrotnie przy różnej liczbie neuronów w warstwie ukrytej: 2, 3 oraz 15.

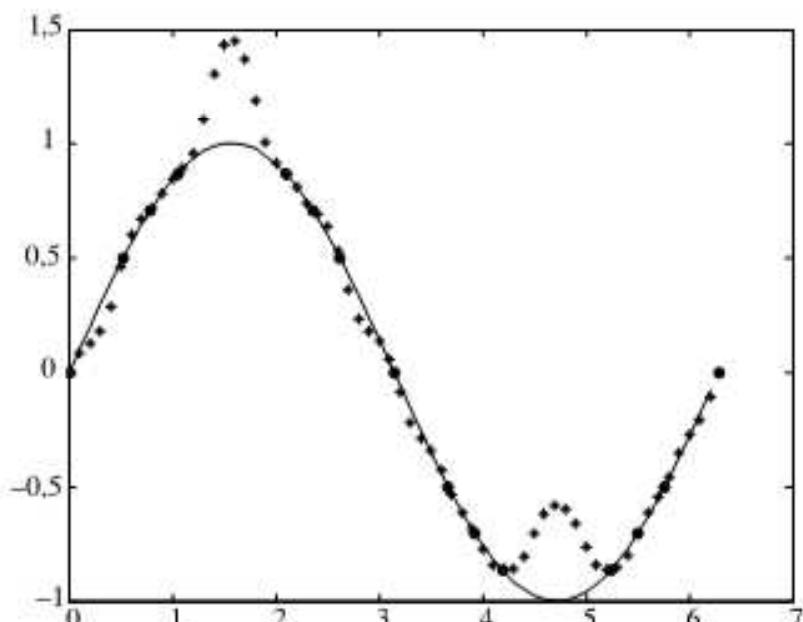


Rys. 6.23. Działanie sieci przy zbyt małej liczbie neuronów — 2 neurony w warstwie ukrytej

Ciąg testowy tworzymy poprzez dyskretyzację przedziału $[0, 2\pi]$ z krokiem 0,1. Rysunki 6.23, 6.24 i 6.25 ilustrują wynik działania sieci, czyli zdolności generalizacji przy podawaniu na jej wejście ciągu testowego. Na każdym z tych rysunków przedstawiono wykres funkcji $f(x) = \sin(x)$, punkty ciągu uczącego podanego w tabeli 6.4 oraz punkty odzwierciedlające sygnały wyjściowe sieci po podaniu na jej wejście ciągu testowego.



Rys. 6.24. Działanie sieci przy odpowiedniej liczbie neuronów — 3 neurony w warstwie ukrytej



Rys. 6.25. Działanie sieci przy zbyt dużej liczbie neuronów — 15 neuronów w warstwie ukrytej

Jak można zauważyć na rysunku 6.23, zbyt mała liczba neuronów w stosunku do liczby próbek uczących powoduje, że sieć nie jest w stanie aproksymować funkcji. Natomiast rysunek 6.25 obrazuje fakt, że zbyt duża liczba neuronów powoduje nadmierne dopasowanie się sieci do ciągu uczącego. Błąd na wyjściu sieci maleje do 0 podczas uczenia, natomiast gwałtownie wzrasta w przypadku ciągu testującego. Naturalne więc wydaje się pytanie: jak określić liczbę neuronów ukrytych w sieci? W pewnym stopniu może być nam pomocny tzw. *wymiar Vapnika–Chervonenkisa* (VC). Wymiar VC dla zbioru funkcji oznacza się literą h i definiuje jako maksymalną liczbę wektorów, które

można pogrupować na wszystkie możliwe sposoby, używając funkcji z tego zbioru. Zauważmy, że w przypadku klasyfikacji binarnej, l wektorów możemy podzielić na dwie klasy na wszystkie 2^l możliwych sposobów. Jeżeli znajdziemy zbiór funkcji, który będzie w stanie dokonać takiej klasyfikacji (na wszystkie 2^l sposobów), to wymiar VC dla tego zbioru funkcji będzie wynosił $h = l$. Niech $i_f(\mathbf{x}, \mathbf{w})$ będzie funkcją przyjmującą tylko dwie wartości, tzn.

$$i_f(\mathbf{x}, \mathbf{w}) \in \{-1, 1\}. \quad (6.160)$$

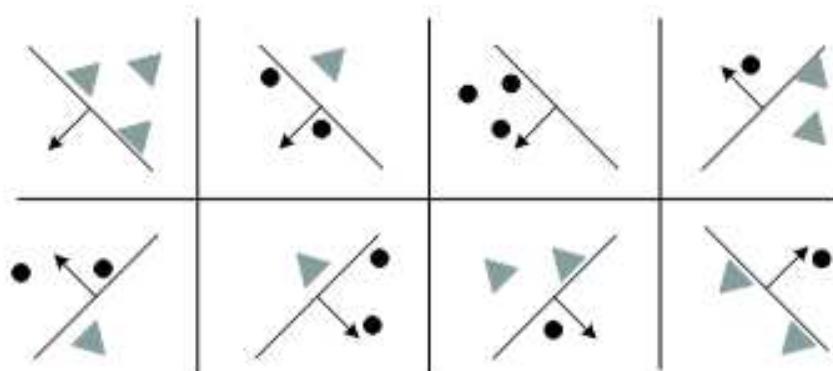
Powyzszy warunek spełnia funkcja opisująca perceptron

$$i_f(\mathbf{x}, \mathbf{w}) = \text{sign}(\mathbf{x}^T \mathbf{w}). \quad (6.161)$$

W przypadku dwuwymiarowym mamy

$$i_f(\mathbf{x}, \mathbf{w}) = \text{sign} \left(\sum_{i=1}^2 w_i x_i + w_0 \right). \quad (6.162)$$

gdzie $\mathbf{x} = [x_1, x_2]^T$ oraz $\mathbf{w} = [w_0, w_1, w_2]^T$. Jak wspominaliśmy, działanie perceptronu o dwóch wejściach ilustruje prosta, która dzieli płaszczyznę na dwie części. Oznacza to, że w zależności od wartości wag tworzy on zbiór funkcji, które mogą wektorom wejściowym przyporządkowywać wartości $+1$ lub -1 , w zależności od tego, po której stronie prostej znajdują się ich współrzędne. Na rysunku 6.26 przedstawiamy wszystkie możliwe podziały płaszczyzny wyznaczone przez perceptron z odpowiednio dobranymi wagami dla trzech wektorów wejściowych. Na rysunku tym strzałkami zaznaczono półprzestrzenie odpowiadające dodatnim wartościom funkcji i_f . W tym przypadku wymiar VC wynosi 3.



Rys. 6.26. Podział płaszczyzny przez perceptron dla trzech wektorów wejściowych

Jeżeli wymiar VC wynosi h , to oznacza, że istnieje co najmniej jeden zbiór h wektorów, które możemy podzielić na wszystkie możliwe sposoby. Nie oznacza to, że właściwość ta musi dotyczyć wszystkich wektorów z danego zbioru. W przypadku funkcji zdefiniowanych zależnością (6.161) w przestrzeni n wymiarowej, wymiar VC wynosi $h = n + 1$. W szczególności dla płaszczyzny jest on równy 3. Z powyższego stwierdzenia można wyciągnąć wniosek, iż wymiar VC wzrasta wraz ze wzrostem liczby wag w neuronie (gdy wzrasta wymiar przestrzeni, wzrasta też liczba wag w neuronie). Jednakże zostało wykazane, że wzrost liczby wag nie zawsze wpływa na wzrost wymiaru VC [108]. W rzeczywistości bardzo trudno jest określić wartość wymiaru VC. Jednakże

liczbę tę należy traktować jako wskazówkę przy określaniu możliwości generalizacyjnych wielowarstwowych sieci neuronowych, co ilustruje zależność [73]

$$Q_G(\mathbf{w}) \leq Q_U(\mathbf{w}) + \Phi\left(\frac{M}{h}, Q_U(\mathbf{w})\right), \quad (6.163)$$

w której M jest liczbą próbek uczących, Q_U oznacza błąd uczenia sieci, Q_G jest błędem generalizacji, natomiast Φ jest pewną funkcją zależną od M , Q_U oraz wymiaru VC. Budując i ucząc sieć neuronową, należy dążyć do tego, aby błąd generalizacji sieci był jak najmniejszy. Pamiętać należy także, że wraz ze wzrostem liczby neuronów w sieci wzrasta czas jej działania. Dlatego powstało wiele metod, które służą do rozbudowy i redukcji struktury sieci. W algorytmach redukujących strukturę sieci usuwa się z sieci neurony lub połączenia (wagi), które nie mają znaczenia lub ich znaczenie jest znikome. Ingerencja w strukturę sieci wykonywana jest zawsze po jej nauczeniu. Określa się błąd sieci dla całego ciągu testującego przed usunięciem wagi, a następnie po jej usunięciu. Jeżeli błąd nie wzrośnie, to oznacza, że zostały usunięte właściwe wagi. Najprostszą metodą jest usuwanie tych wag, które mają wartości mniejsze od zadanego przez nas progu. Można przypuszczać, że mała wartość wagi ma niewielki wpływ na łączne pobudzenie neuronu. Niestety, nie zawsze takie działanie ma sens. Okazuje się bowiem, że niekiedy usunięcie takiej wagi może spowodować gwałtowny wzrost błędu wyjściowego sieci. Dlatego po wyrzuceniu wagi zawsze należy sieć ponownie uczyć, a następnie sprawdzić, czy struktura działa lepiej, czy też usuniętą wcześniej wagę trzeba ponownie wkomponować w strukturę sieci. Przykładem takiego postępowania może być *metoda rozpadu wag*. W algorytmie tym do standardowej postaci funkcji błędu sieci dodaje się człon regularyzacyjny i w efekcie funkcję tę definiuje się następująco:

$$Q = \frac{1}{2} \sum_{i=1}^{N_t} (y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{i=1}^I w_i^2, \quad (6.164)$$

gdzie I jest liczbą wszystkich wag w sieci. Po nauczeniu sieci ze struktury można usunąć te wagi, których wartości są mniejsze niż założony przez nas próg. Powyższa zmiana powoduje jednak, że w wyniku uczenia w strukturze powstaje duża liczba wag o małych wartościach. Dlatego w literaturze (np. [12]) zaproponowano różne modyfikacje wzoru (6.164). Inną metodą redukcji struktury sieci jest *algorytm OBD* (ang. *Optimal Brain Damage*). Ponieważ chcemy usunąć pewne wagi w nauczonej strukturze, powinniśmy oszacować ich wpływ na działanie sieci. Oznaczmy błąd sieci po usunięciu wag przez $Q(\mathbf{w})$, natomiast przed usunięciem, ale po nauczeniu sieci — przez $Q(\mathbf{w}^*)$. Jak pamiętamy, wyprowadzając algorytm wstecznej propagacji błędów, aproksymowaliśmy funkcję błędu za pomocą szeregu Taylora, wykorzystując zależność (6.95). Ponieważ redukcji wag dokonujemy, gdy sieć neuronowa jest już nauczona, więc możemy przyjąć, że funkcja błędu osiągnęła minimum. Zgodnie z tym składowe wektora gradientu funkcji Q przyjęłyby wartości zerowe. Chcąc oszacować wartość różnicy błędów przed i po redukcji struktury, pomijając wektor gradientu i obcinając rozwinięcie szeregu do trzeciego wyrazu, otrzymujemy

$$Q(\mathbf{w}) - Q(\mathbf{w}^*) \approx 0,5(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w}^*)(\mathbf{w} - \mathbf{w}^*). \quad (6.165)$$

Wyznaczenie hesjanu we wzorze (6.165) jest kłopotliwe ze względu na dużą liczbę wag neuronów w sieci. Dlatego twórcy metody OBD przyjęli, że największy wpływ na wartość hesjanu mają jego elementy diagonalne. Stąd miarę wrażliwości dla j -ej wagi w i -tym neuronie zdefiniowano jako

$$S_{ij} = \frac{1}{2} \frac{\partial^2 Q}{\partial w_{ij}^2} w_{ij}^2. \quad (6.166)$$

Algorytm OBD składa się więc z następujących etapów:

1. Dobór wstępny struktury sieci i jej uczenie.
2. Obliczenie współczynnika wrażliwości dla wszystkich wag zgodnie ze wzorem (6.166) i usunięcie tych, dla których wartość S_{ij} jest niewielka.
3. Ponowne uczenie sieci.

Macierz hesjanu wag jest macierzą niediagonalną i algorytm OBD może spowodować usunięcie z sieci istotnych wag. Dlatego w literaturze zaproponowano tzw. *metodę OBS* (ang. *Optimal Brain Surgeon*), w której uwzględnia się wszystkie składowe hesjanu. Najpierw obliczamy współczynniki

$$s_j = \frac{1}{2} \frac{w_j^2}{\mathbf{H}_{jj}^{-1}}. \quad (6.167)$$

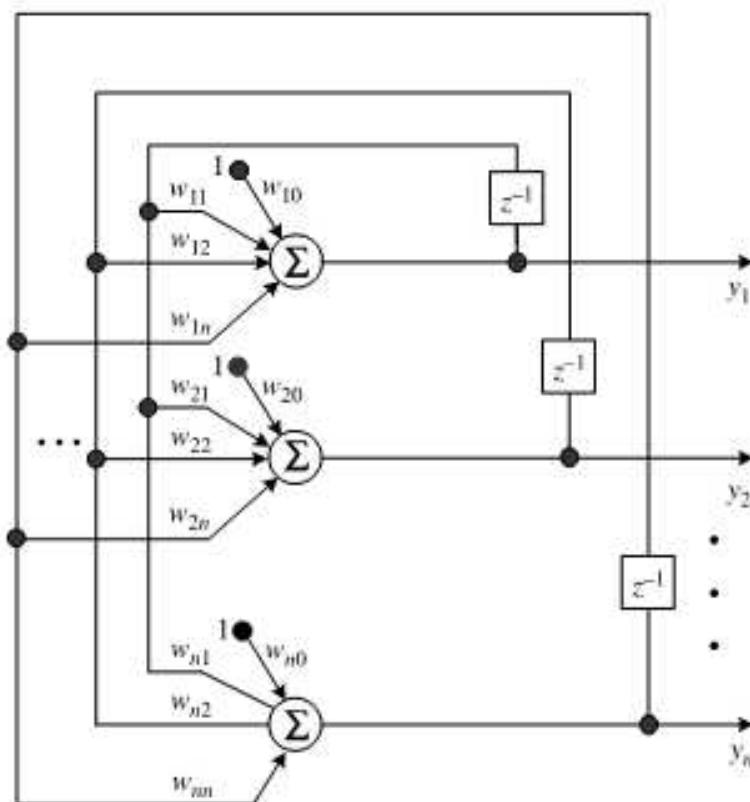
Następnie obcięciu ulegają wagi o najmniejszych wartościach współczynnika (6.167). Korekty, o które należy zmodyfikować wagi, wyznacza się według zależności

$$\Delta \mathbf{w} = -\frac{w_j}{\mathbf{H}_{jj}^{-1}} \mathbf{H}^{-1} \cdot \mathbf{i}_j, \quad (6.168)$$

w której \mathbf{i}_j jest wektorem składającym się z samych zer i jedynki na j -ej pozycji.

6.4. Sieci rekurencyjne

We wszystkich dotychczas omawianych sieciach nie rozważaliśmy przypadku, w którym sygnał otrzymany na wyjściu sieci trafiał powtórnie na jej wejście. Taki obieg sygnału nazywamy *sprzężeniem zwrotnym*. Struktury, w których to zjawisko występuje, nazywamy *sieciami rekurencyjnymi*. Jednorazowe pobudzenie struktury ze sprzężeniem zwrotnym może generować całą sekwencję nowych zjawisk i sygnałów, ponieważ sygnały z wyjścia sieci trafiają ponownie na jej wejście, generując nowe sygnały, aż do ustabilizowania się sygnałów wyjściowych. Takiemu przebiegowi towarzyszą często tłumienia, oscylacje, gwałtowne narastanie lub opadanie sygnałów. Przedstawimy architektury oraz zwięzle omówimy działanie najbardziej znanych sieci rekurencyjnych, a mianowicie sieci Hopfielda, Hamminga, RTRN, Elmana i BAM (ang. *Bidirectional Associative Memory*). Warto wspomnieć, że sieci rekurencyjne mają zastosowanie jako pamięci asocjacyjne. Przykładowo sieci Hopfielda mogą pełnić funkcję pamięci typu autoasocjacyjnego, natomiast sieci Hamminga oraz sieci typu BAM są przykładami pamięci typu heteroasocjacyjnego (kojarzenie dwóch różnych wektorów).



Rys. 6.27. Schemat sieci Hopfielda

6.4.1. Sieć Hopfielda

Na rysunku 6.27 przedstawiono schemat sieci Hopfielda. Jest to sieć jednowarstwowa o regularnej budowie, składająca się z wielu neuronów połączonych każdy z każdym. Nie istnieją sprzężenia zwrotne obejmujące ten sam neuron. Oznacza to, że sygnał wyjściowy danego neuronu nie trafia na jego wejście, a więc wartości wag w_{ii} są równe 0. Wagi w tej sieci są symetryczne, tzn. waga w_{kj} łącząca neuron k z neuronem j jest równa wadze w_{jk} łączącej neuron j z neuronem k . Sieć Hopfielda podczas uczenia modyfikuje swoje wagi w_{kj} w zależności od wartości wektora uczącego \mathbf{x} . W trybie odtworzeniowym wagi nie ulegają modyfikacjom, natomiast sygnał wejściowy pobudza sieć, która poprzez sprzężenie zwrotne wielokrotnie przyjmuje na swoje wejście sygnał wyjściowy, aż do ustabilizowania odpowiedzi. Jeżeli przyjmiemy, że funkcja aktywacji neuronów jest typu signum, to działanie sieci w kroku t możemy zapisać jako

$$y_k(t) = \operatorname{sgn} \left(\sum_{j=1, j \neq k}^n w_{kj} y_j(t-1) + \theta_k \right), \quad k = 1, \dots, N, \quad (6.169)$$

przy czym $y_j(0) = x_j$. Sygnał na wyjściu będzie zmieniał się tak długo, aż w kroku $t-1$ będzie równy sygnałowi w kroku t , a więc $y_k(t) = y_k(t-1)$ dla wszystkich N neuronów wchodzących w skład sieci. Wykażemy teraz, że jeżeli wagi w sieci Hopfielda są symetryczne, to sieć ta zawsze się stabilizuje. Oznaczmy wyjście części liniowej k -tego neuronu w chwili $t+1$ przez $s_k(t+1)$. Możemy zapisać

$$s_k(t+1) = \sum_{j=1, j \neq k}^n w_{kj} y_j(t) + \theta_k, \quad (6.170)$$

gdzie θ_k jest wartością progową neuronu. Funkcję aktywacji można zdefiniować następująco:

$$y_k(t+1) = \operatorname{sgn}(s_k(t+1)). \quad (6.171)$$

Sieć jest ustabilizowana, jeżeli zachodzi

$$y_k(t) = y_k(t-1) \quad (6.172)$$

dla każdego neuronu. Przyjmując funkcję aktywacji wyrażoną wzorem (6.171), zależność (6.172) zapiszemy jako

$$y_k(t) = \operatorname{sgn}(s_k(t-1)). \quad (6.173)$$

Stan energetyczny sieci wyraża funkcja Lapunowa w postaci

$$E = -\frac{1}{2} \sum_j \sum_{\substack{k \\ j \neq k}} y_j y_k w_{kj} - \sum_k \theta_k y_k. \quad (6.174)$$

Zauważmy, że funkcja energetyczna (6.174) jest ograniczona od dołu, przy czym wagи i wartości progowe są stałe. Łatwo sprawdzić, że zmiana energii

$$\Delta E = -\Delta y_k \left(\sum_{j \neq k} y_j w_{kj} + \theta_k \right) \quad (6.175)$$

jest zawsze ujemna, gdy sygnał y_k zmienia się według zależności (6.171). Stąd funkcja energetyczna E jest funkcją malejącą w kolejnych krokach t . Dlatego też możemy być pewni, że osiągnie minimum, w którym sieć będzie stabilna. Omówimy teraz sposób doboru wag w sieci Hopfielda. Jedną z metod uczenia sieci Hopfielda jest *uogólniona reguła Hebb'a*. Zgodnie z tą regułą wag modyfikowane są według zależności

$$w_{kj} = \frac{1}{N} \sum_{i=1}^M x_k^i x_j^i, \quad (6.176)$$

w której $\mathbf{x}^i = [x_1^i, \dots, x_n^i]$, $i = 1, \dots, M$. Niestety, nie daje ona najlepszych efektów, gdyż jak można wykazać, pojemność sieci (maksymalna liczba wzorców, którą sieć jest w stanie zapamiętać) uczonej za pomocą tej reguły stanowi zaledwie 13,8% liczby neuronów. Dlatego w praktyce stosuje się częściej metodę pseudoinwersji. Niech \mathbf{X} będzie macierzą M wektorów uczących, tzn. $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^M]$. Zakłada się, że celem uczenia sieci jest taki dobór wag, aby po podaniu na jej wejście sygnału \mathbf{x} , na wyjściu powstał ten sam sygnał, tzn.

$$\mathbf{W}\mathbf{X} = \mathbf{X}, \quad (6.177)$$

gdzie \mathbf{W} oznacza macierz wag o wymiarach $n \times n$. Rozwiążanie układu równań (6.177) jest następujące:

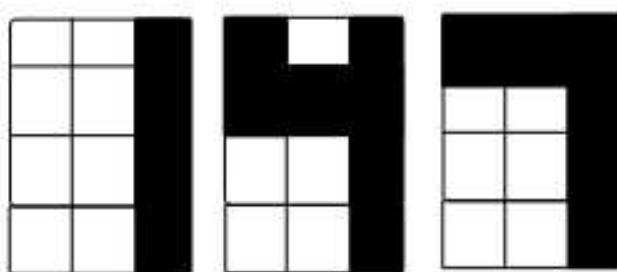
$$\mathbf{W} = \mathbf{X}\mathbf{X}^+, \quad (6.178)$$

gdzie znak $+$ oznacza pseudoinwersję. Jeżeli przyjmiemy, że wektory uczące są liniowo niezależne, to równanie (6.178) przybierze postać

$$\mathbf{W} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T. \quad (6.179)$$

Przykład 6.6

Przeprowadzimy teraz próbę nauczenia sieci Hopfielda rozpoznawania trzech cyfr: 1, 4, 7. Doświadczenie to przeprowadzimy w środowisku Matlab. Na podstawie wzorców przedstawionych na rysunku 6.28 tworzymy ciąg uczący (białe pola oznaczamy przez -1 czarne przez 1)



Rys. 6.28. Wzorce uczące do przykładu 6.6

$$\mathbf{x}(1) = [-1 -1 1 -1 -1 1 -1 1]$$

$$\mathbf{x}(2) = [1 -1 1 1 1 -1 -1 1]$$

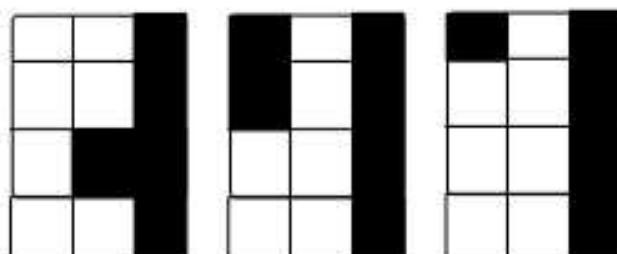
$$\mathbf{x}(3) = [1 1 1 -1 -1 1 -1 1]$$

Następnie uczymy sieć Hopfielda za pomocą powyższych wzorców. Aby sprawdzić poprawność działania nauczonej sieci, na jej wejście podajemy kolejno sygnały uczące. Jak wynika z tabeli 6.5, sieć perfekcyjnie rozwiązała zagadnienie asocjacji dla niezaszumionych wektorów uczących.

Tabela 6.5. Wynik działania sieci Hopfielda dla niezaszumionych wzorców

Wejście	Wyjście
$-1 -1 1 -1 -1 1 -1 1$	$-1 -1 1 -1 -1 1 -1 1$
$1 -1 1 1 1 -1 -1 1$	$1 -1 1 1 1 -1 -1 1$
$1 1 1 -1 -1 1 -1 1$	$1 1 1 -1 -1 1 -1 1$

Teraz zaszumimy wzorce (rys. 6.29) i zobaczymy, jaka będzie odpowiedź sieci. W przypadku zaszumienia sygnały wejściowe \mathbf{x}_z są postaci



Rys. 6.29. Zaszumione wzorce uczące do przykładu 6.6

$$\mathbf{x}_z(1) = [-1 -1 1 -1 1 -1 1 -1]$$

$$\mathbf{x}_z(2) = [1 -1 1 1 -1 1 -1 1]$$

$$\mathbf{x}_z(3) = [1 -1 1 -1 -1 1 -1 1]$$

Efekt zaszumienia przedstawia tabela 6.6.

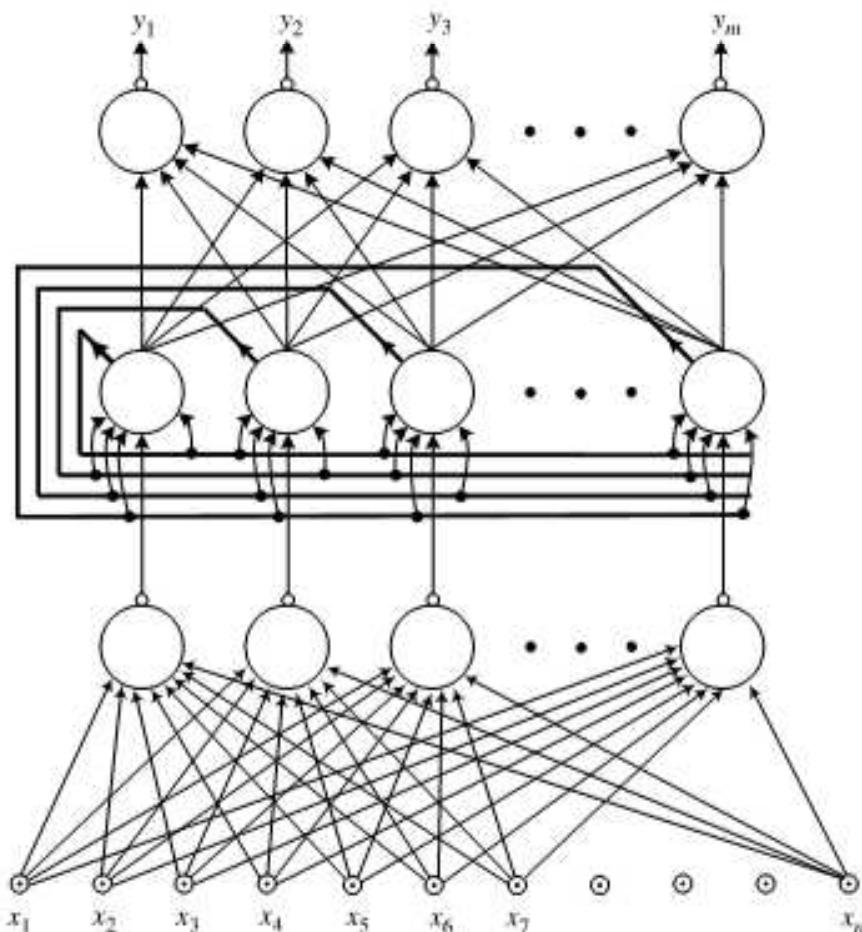
Tabela 6.6. Wynik działania sieci Hopfielda dla zaszumianych wzorców

Wejście	Wyjście	Liczba iteracji
-1 -11 -1 -11 -111 -1 -11	-1 -11 -1 -11 -1 -11 -1 -11 (sieć rozpoznała cyfrę 1)	2
1 -111 -11 -1 -11 -1 -11	1 -11111 -1 -11 -1 -11 (sieć rozpoznała cyfrę 4)	2
1 -11 -1 -11 -1 -11 -1 -11	-0,0571 0,0571 1 -1 -11 -1 -11 -1 -11	12

Dla trzeciego zaszumionego wektora wyjście sieci po 12 iteracjach jeszcze się nie ustabilizowało, co oznacza, że sieć nie mogła rozpoznać znieksztalconej próbki.

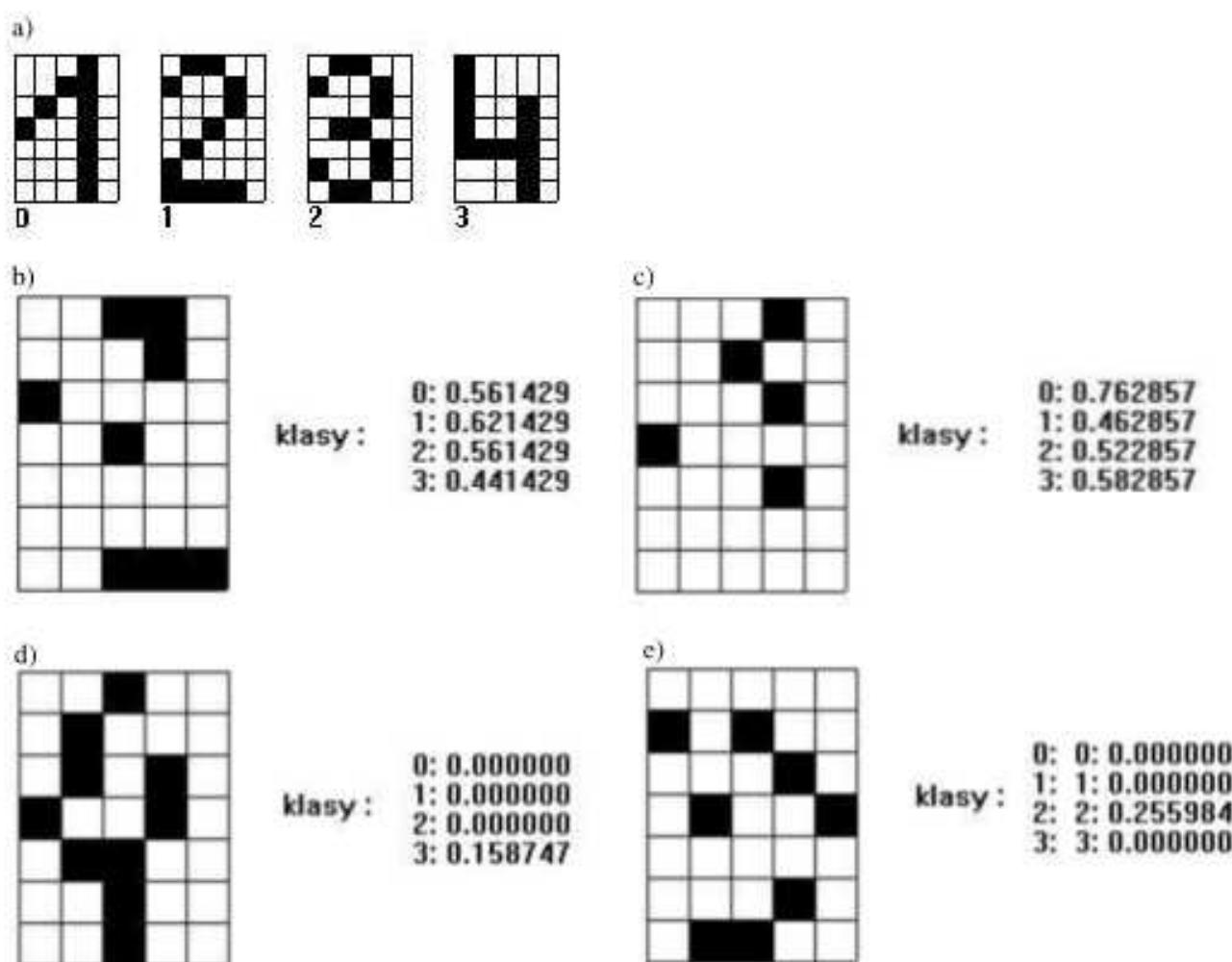
6.4.2. Sieć Hamminga

Przedstawiona na rysunku 6.30 struktura sieci Hamminga jest siecią trójwarstwową, stosowaną do klasyfikacji obrazów. W warstwie pierwszej znajduje się p neuronów, które wyznaczają odległość Hamminga między wektorem wejściowym a każdym z p wektorów wzorcowych zakodowanych w wagach tej warstwy. Druga warstwa nosi nazwę MAXNET. Jest ona warstwą odpowiadającą sieci Hopfielda, której działanie omówiliś-



Rys. 6.30. Struktura sieci Hamminga

my wcześniej. W warstwie tej dodaje się jednak sprzężenia zwrotne obejmujące ten sam neuron. Wagi w tych sprzężeniach są równe 1. Wartości wag pozostałych neuronów tej warstwy dobiera się tak, aby działały hamująco, np. przyjmując ujemne ich wartości. W ten sposób w warstwie MAXNET następuje wygaszenie wszystkich wyjść oprócz tego, które było najsilniejsze w warstwie pierwszej. Neuron tej warstwy utożsamiany ze zwycięzcą, poprzez wagi neuronów wyjściowych o liniowej funkcji aktywacji, odtworzy wektor wyjściowy skojarzony z wektorem zakodowanym w warstwie pierwszej.



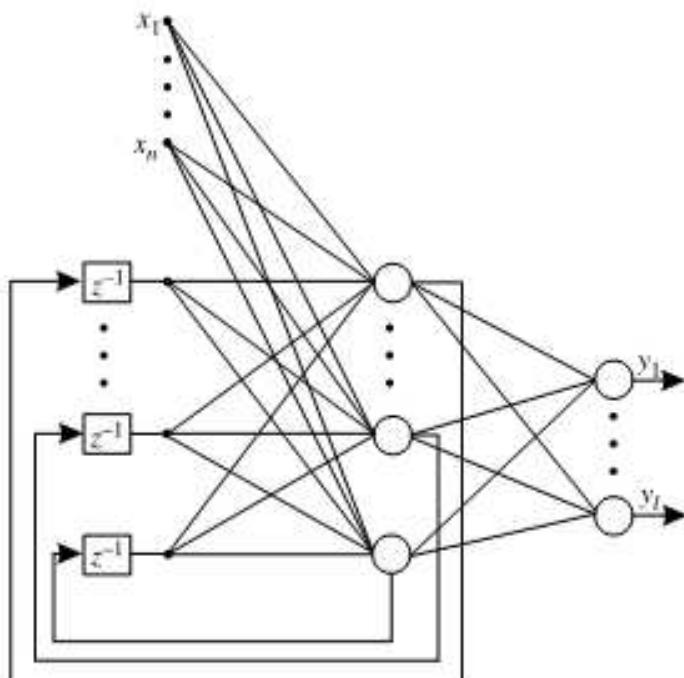
Rys. 6.31. Ilustracje do przykładu 6.7

Przykład 6.7

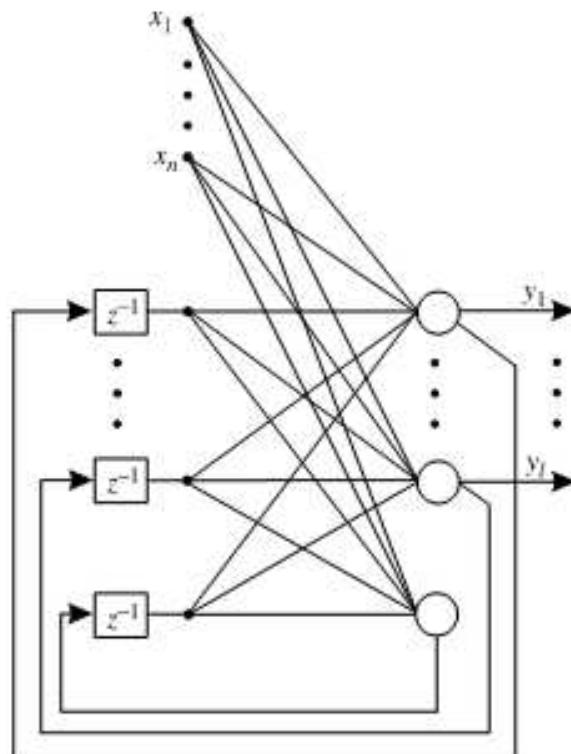
W celu zaprezentowania działania sieci Hamminga utworzyliśmy ciąg uczący w postaci binarnego zapisu graficznej reprezentacji kolejnych cyfr: 1, 2, 3 i 4 (rys. 6.31a). W pierwszej fazie sieć uczyła, wykorzystując każdy z tych wzorców. Następnie na jej wejście podawano wektory, które reprezentowały zaszumione sygnały wzorcowe. Rezultaty przedstawiono na rysunkach 6.31b, 6.31c, 6.31d, 6.31e. W symulacjach wykorzystano program NetLab [271], który klasy odpowiadające cyfrom od 1 do 4 numeruje od 0 do 3. Liczby po prawej stronie ilustracji oznaczają stany neuronów wyjściowych dla każdej z klas wzorcowych. Jak można zauważyć, we wszystkich przypadkach sieć sklasyfikowała zaszumione obrazy prawidłowo, jeśli za właściwą miarę można uznać wrażenie podobieństwa odniesione za pomocą ludzkiego oka.

6.4.3. Sieci wielowarstwowe ze sprzężeniem zwrotnym

W literaturze opisano różne struktury wielowarstwowych sieci neuronowych ze sprzężeniem zwrotnym. Najczęściej stosowane są sieci Elmana oraz sieci typu RTRN (ang. *Real Time Recurrent Network*). Na rysunku 6.32 pokazano schemat sieci Elmana, której nazwa pochodzi od jej pomysłodawcy. W strukturze tej można wyróżnić neurony ze



Rys. 6.32. Schemat sieci Elmana



Rys. 6.33. Schemat sieci RTRN

sprzężeniem zwrotnym wchodzące w skład warstw ukrytych. Każdy z neuronów warstwy ukrytej przetwarza sygnały wejściowe zewnętrzne oraz sygnały będące rezultatem działania sprzężenia zwrotnego. Sygnały z warstwy wyjściowej nie są poddawane operacji sprzężenia zwrotnego. Inną, znaną strukturą jest przedstawiona na rysunku 6.33 sieć RTRN. W przeciwieństwie do sieci Elmana, sprzężenie zwrotne obejmuje zarówno sygnały wyjściowe sieci, jak i neurony ukryte. Sieci Elmana oraz RTRN można uczyć za pomocą algorytmów gradientowych, które przybierają bardziej skomplikowaną postać niż w przypadku uczenia sieci bez sprzężeń zwrotnych. Wielowarstwowe sieci rekurencyjne mają zastosowanie do modelowania ciągów czasowych oraz do identyfikacji obiektów dynamicznych.

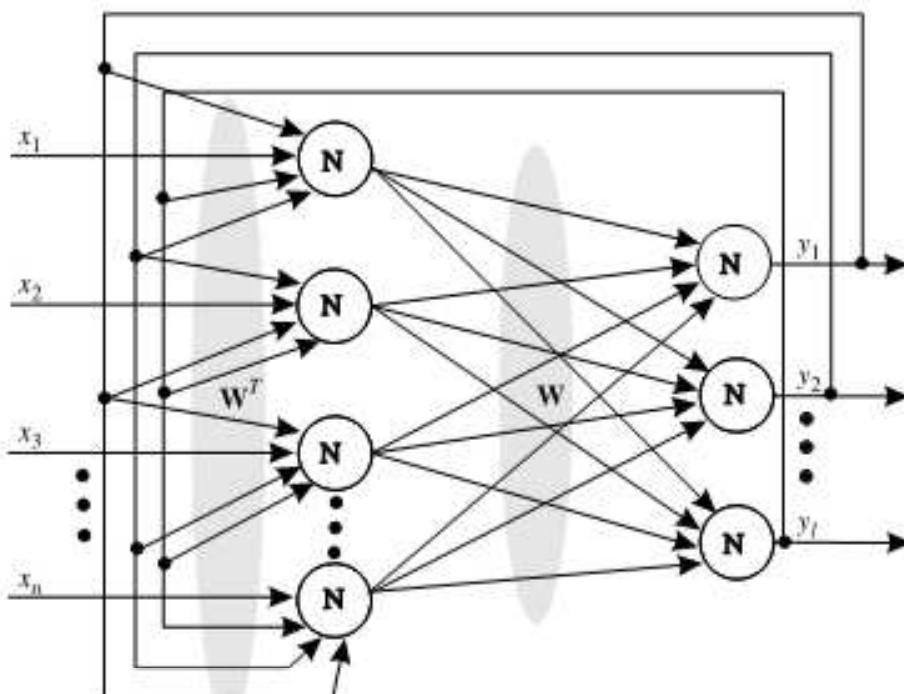
6.4.4. Sieć BAM

Sieć BAM jest siecią rekurencyjną, która umożliwia zapamiętanie zbioru par wektorów wzajemnie skojarzonych ze sobą. Schemat tej struktury przedstawiamy na rysunku 6.34. Jest to sieć dwukierunkowa. Sygnały otrzymane na wyjściu neuronów warstwy pierwszej przekazywane są poprzez wagę tworzącą macierz \mathbf{W} na wejścia neuronów znajdujących się w warstwie drugiej. Sygnały wyjściowe neuronów tej warstwy zostają skierowane

poprzez macierz \mathbf{W}^T ponownie na wejścia neuronów w pierwszej warstwie. Cykl ten powtarza się aż do ustalenia stanu równowagi sieci. Macierz wag \mathbf{W} jest dobierana na podstawie następującej zależności:

$$\mathbf{W} = \sum_{i=1}^M \mathbf{x}_i^T \mathbf{y}_i, \quad (6.180)$$

w której M oznacza liczbę wektorów uczących, $\mathbf{x} = [x_1, \dots, x_n]^T$ jest wektorem sygnałów wejściowych, natomiast $\mathbf{y} = [y_1, \dots, y_l]^T$ wektorem sygnałów wyjściowych.



Rys. 6.34. Schemat sieci BAM

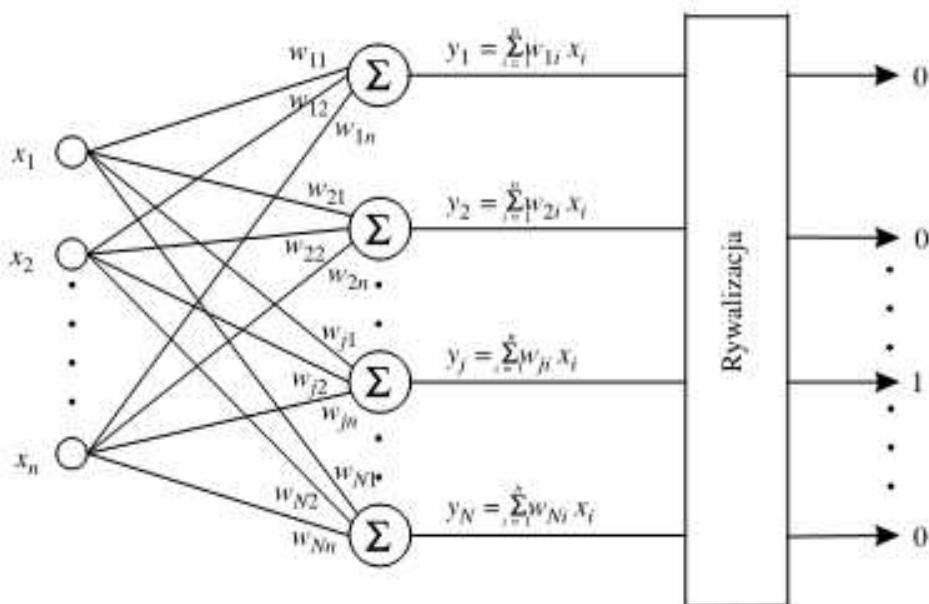
Funkcje aktywacji w neuronach są unipolarne lub bipolarne. W procesie uczenia wektory uczące unipolarne są zamieniane na bipolarne, ponieważ sieć uczona za pomocą sygnałów bipolarnych wykazuje lepsze właściwości w trybie odtworzeniowym. Ponadto często stosuje się różne modyfikacje wzoru (6.180), aby poprawić działanie sieci BAM w fazie odtworzeniowej.

6.5. Sieci samoorganizujące z konkurencją

Sieci zaprezentowane w podrozdziale 6.3 były uczone z wykorzystaniem algorytmów z nauczycielem. Oznacza to, że znaliśmy wzorcową odpowiedź struktury na sygnał wejściowy. Teraz pokażemy sieci samoorganizujące. Uczenie ich jest nazywane *uczeniem nienadzorowanym* lub *uczeniem bez nauczyciela*. Jak można przypuszczać, ciąg uczący składa się jedynie z wartości wejściowych, bez wzorcowego (zadanego) sygnału wyjściowego. Sieci samoorganizujące mają nieskomplikowaną budowę, a do zrozumienia ich działania wystarcza niewielka znajomość matematyki. Mimo swej prostoty sieci te mają szerokie zastosowanie między innymi w zadaniach grupowania danych.

6.5.1. Sieci typu WTA

Pierwszą siecią samoorganizującą, którą opiszemy, będzie sieć uczona za pomocą algorytmu WTA (ang. *Winner Takes All*). Na rysunku 6.35 przedstawiono schemat sieci



Rys. 6.35. Sieć samoorganizująca typu WTA

tego typu. Sygnał $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ podany na wejście sieci trafia na wejścia wszystkich N neuronów. Tutaj następuje wyznaczenie miary podobieństwa (odległości) sygnału wejściowego \mathbf{x} do wszystkich wektorów wag $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$, $i = 1, \dots, N$. Najczęściej stosujemy miarę euklidesową

$$d(\mathbf{x}, \mathbf{w}_i) = \|\mathbf{x} - \mathbf{w}_i\| = \sqrt{\sum_{j=1}^n (x_j - w_{ij})^2} \quad (6.181)$$

lub iloczyn skalarny

$$d(\mathbf{x}, \mathbf{w}_i) = 1 - \mathbf{x} \circ \mathbf{w}_i = 1 - \|\mathbf{x}\| \|\mathbf{w}_i\| \cos(\mathbf{x}, \mathbf{w}_i). \quad (6.182)$$

Neuron charakteryzujący się najmniejszą odległością wektora wag od sygnału wejściowego na swym wyjściu przyjmuje wartość 1, a pozostałe neurony przyjmują wartość 0 (stąd nazwa: „zwycięzca bierze wszystko”). Zatem dla j -ego neuronu-zwycięzcy możemy zapisać

$$d(\mathbf{x}, \mathbf{w}_j) = \min_{1 \leq i \leq N} d(\mathbf{x}, \mathbf{w}_i), \quad (6.183)$$

gdzie d jest jedną z powyższych miar podobieństwa. W procesie uczenia tylko wagi zwycięzcy są modyfikowane, zgodnie z regułą

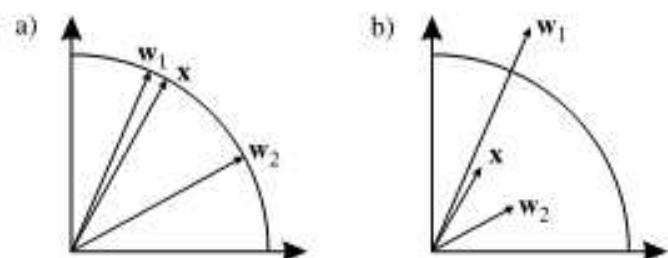
$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t)[\mathbf{x}(t) - \mathbf{w}_j(t)], \quad (6.184)$$

przy czym η jest wielkością kroku modyfikacji wag. W sieciach samoorganizujących warto dokonać normalizacji sygnałów wejściowych (norma wektora \mathbf{x} jest równa 1), gdyż

zapewni to spójny podział przestrzeni danych. Normalizacji można dokonać poprzez redefinicję składowych wektora \mathbf{x} w sposób następujący:

$$x'_i = \frac{x_i}{\sqrt{\sum_{i=1}^n x_i^2}}. \quad (6.185)$$

Znormalizowanie wektora uczącego \mathbf{x} powoduje normalizację wektora wag w trakcie procesu uczenia. Ponadto na działanie neuronu nie będzie miała wpływu długość wektora wag, a jedynie kosinus kąta pomiędzy nim a wektorem wejściowym \mathbf{x} (miara euklidesowa i iloczyn skalarny są w tym przypadku równe).



Rys. 6.36. a) Znormalizowane wektory wag i wejścia;
b) wektory bez normalizacji

Na rysunku 6.36a przedstawiono wektory wag i wejścia znormalizowanych, a na rysunku 6.36b wektory bez normalizacji o tych samych kierunkach. Łatwo zauważać, że jeżeli wektory są znormalizowane, to iloczyn skalarny $\mathbf{x} \circ \mathbf{w}_1 = \|\mathbf{x}\| \|\mathbf{w}_1\| \cos \alpha$ jest większy niż iloczyn skalarny wektorów \mathbf{x} i \mathbf{w}_2 (ponieważ $\cos \alpha$ rośnie wraz ze zmniejszaniem się kąta α). Na rysunku 6.36b także iloczyn skalarny \mathbf{x} i \mathbf{w}_1 jest większy niż \mathbf{x} i \mathbf{w}_2 , chociaż patrząc na schemat, mamy wrażenie, że wektory \mathbf{x} i \mathbf{w}_2 są bardziej do „siebie podobne”.

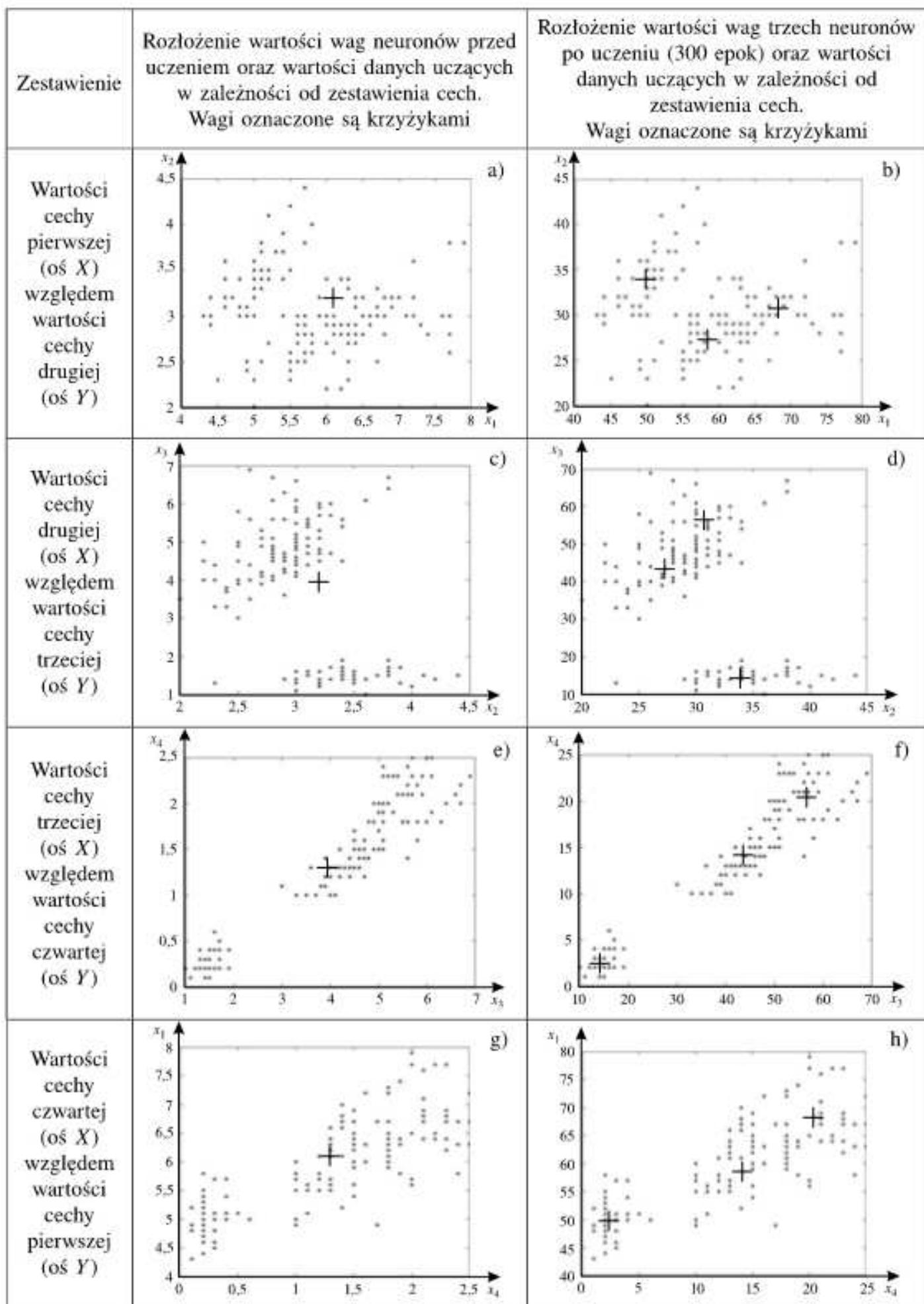
Działanie sieci samoorganizującej sprowadza się do podzielenia M wzorców uczących na N klas (N oznacza liczbę neuronów w sieci). Po jej nauczeniu, każdy neuron reprezentuje inną klasę, wektory wag w stają się środkami (centrami) klas, które wyróżniła uczona przez nas struktura. Możemy pokusić się o zdefiniowanie miary błędu klasyfikacji dla sieci samoorganizującej. Możemy stwierdzić, że zadaniem sieci jest taki dobór wartości wag, aby dla wektora wejściowego \mathbf{x} , który należy do i -tej klasy (i -ty neuron powinien być zwycięzcą), jego odległość od i -tego wektora wag była jak najmniejsza. Ogólnie: powinniśmy tak modyfikować wagi, aby dla wszystkich M wzorców uczących \mathbf{x} , ich odległości od odpowiednich środków klas były jak najmniejsze. Naszym zadaniem jest minimalizacja błędu kwantyzacji danego wzorem

$$Q = \frac{1}{M} \sum_{t=1}^M \|\mathbf{w}^*(t) - \mathbf{x}(t)\|, \quad (6.186)$$

w którym $\mathbf{w}^*(t)$ jest wagą neuronu-zwycięzcą przy podaniu na wejście wektora $\mathbf{x}(t)$.

Przykład 6.8

Zastosujemy algorytm WTA do rozwiązania problemu kwiatu irysa. Na podstawie czterech jego cech: długości i szerokości płatka oraz długości i szerokości liścia, określmy, do którego z trzech gatunków należy. Jak można się domyślać, każdy wejściowy wektor uczący składa się z czterech składowych ($n = 4$). Wynika z tego, że każdy neuron musi



Rys. 6.37. Ilustracja do przykładu 6.8

mieć cztery wejścia, a więc także cztery wag. Liczba rozróżnianych gatunków sugeruje nam trzy neurony potrzebne w sieci ($N = 3$). Zatem łącznie w procesie uczenia sieci będziemy dobierać wartości 12 wag. Wszystkich wektorów uczących mamy 120.

Na rysunku 6.37 przedstawiono rozłożenie wartości wag przed uczeniem i po uczeniu dla różnych kombinacji cech. Przed uczeniem wartości początkowe wag na poszczególnych rysunkach są sobie równe. Po zakończeniu uczenia wartości wag można przedstawić w postaci następującej macierzy:

$$\mathbf{W} = \begin{bmatrix} 58,3060 & 27,2904 & 43,2095 & 13,969 \\ 50,0810 & 34,3020 & 14,6303 & 2,4686 \\ 67,7765 & 30,3668 & 56,0270 & 20,0741 \end{bmatrix}.$$

Na podstawie rozkładu wartości cech (rys. 6.37) możemy wnioskować, jakie ich wartości determinują daną klasę. Aby przetestować sieć, postanowiliśmy podać na jej wejście 150 wektorów testujących (w tym 120 wektorów uczących). Każdy wektor testujący składał się z 4 sygnałów wejściowych oraz z sygnału wzorcowego informującego, do jakiego gatunku należy dany wzorzec. Wyniki testowania zaprezentowano w tabeli 6.7.

Tabela 6.7. Wyniki testowania w przykładzie 6.8

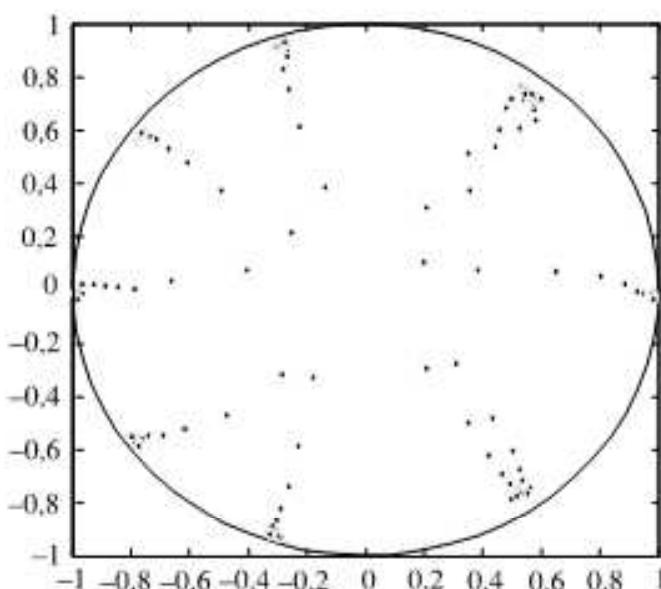
Numer próbki	1 ... 50	51, ... 53	52, ... 54	77, ... 78	79 100	101	102	103 ... 106	107	108 ... 113	114	115	120 ... 127, 128	121, ... 126, 129 ... 138	139, ... 143	142, ... 144	140
Numer neuronu-zwycięzcy	2	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	3
Wartość sygnału wzorcowego	1	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
Błąd tak/nie	N	T	N	T	N	N	T	N	T	N	T	N	T	N	T	N	N

W pierwszym wierszu tabeli znajdują się numery kolejnych próbek. Drugi wiersz zawiera informację o tym, który neuron zwyciężył po podaniu na wejście sieci samoorganizującej próbek o konkretnych numerach. W trzecim wierszu zamieściliśmy informację, znajdującą się w ciągu testowym, do której klasy należy dana próbka. Czytelnik, analizując dane z tabeli, może stwierdzić, że nauczona przez nas struktura nie działa prawidłowo, gdyż numery neuronów zwycięskich oraz numery klas, do których powinna należeć dana próbka, nie pokrywają się. Nic bardziej mylnego. Podczas uczenia nie „mówiliśmy” sieci, który neuron odpowiada za dany gatunek (klasę). Dlatego neuron 2 należy traktować jako reprezentanta gatunku 1, neuron 1 jako reprezentanta gatunku 2, neuron trzeci zaś należy utożsamiać z gatunkiem 3. Teraz dopiero możemy wyciągnąć

prawidłowe wnioski. Mianowicie, pierwszy gatunek irysa sieć rozpoznaje bezbłędnie (dla pierwszych 50 próbek sieć cały czas wskazuje tę samą klasę — wygrywa neuron 2), natomiast błędy pojawiają się w przypadku gatunku drugiego i trzeciego.

Przykład 6.9

Kolejnym problemem jest zadanie rozpoznawania okręgu oraz kwadratu. Najpierw na płaszczyźnie zdefiniowaliśmy okrąg, a następnie ustawiłyśmy wartości początkowe wag dziesięciu neuronów w jego środku. Ciąg uczący został wygenerowany na podstawie wybranych punktów leżących na okręgu. W czasie 250 epok działania algorytmu WTA ($n = 2$, $N = 10$) rysowaliśmy co 25 epok przemieszczanie się wag. W rezultacie wartości wag neuronów przesunęły się na brzeg okręgu, tworząc „ścieżki” pokazane na rysunku 6.38.



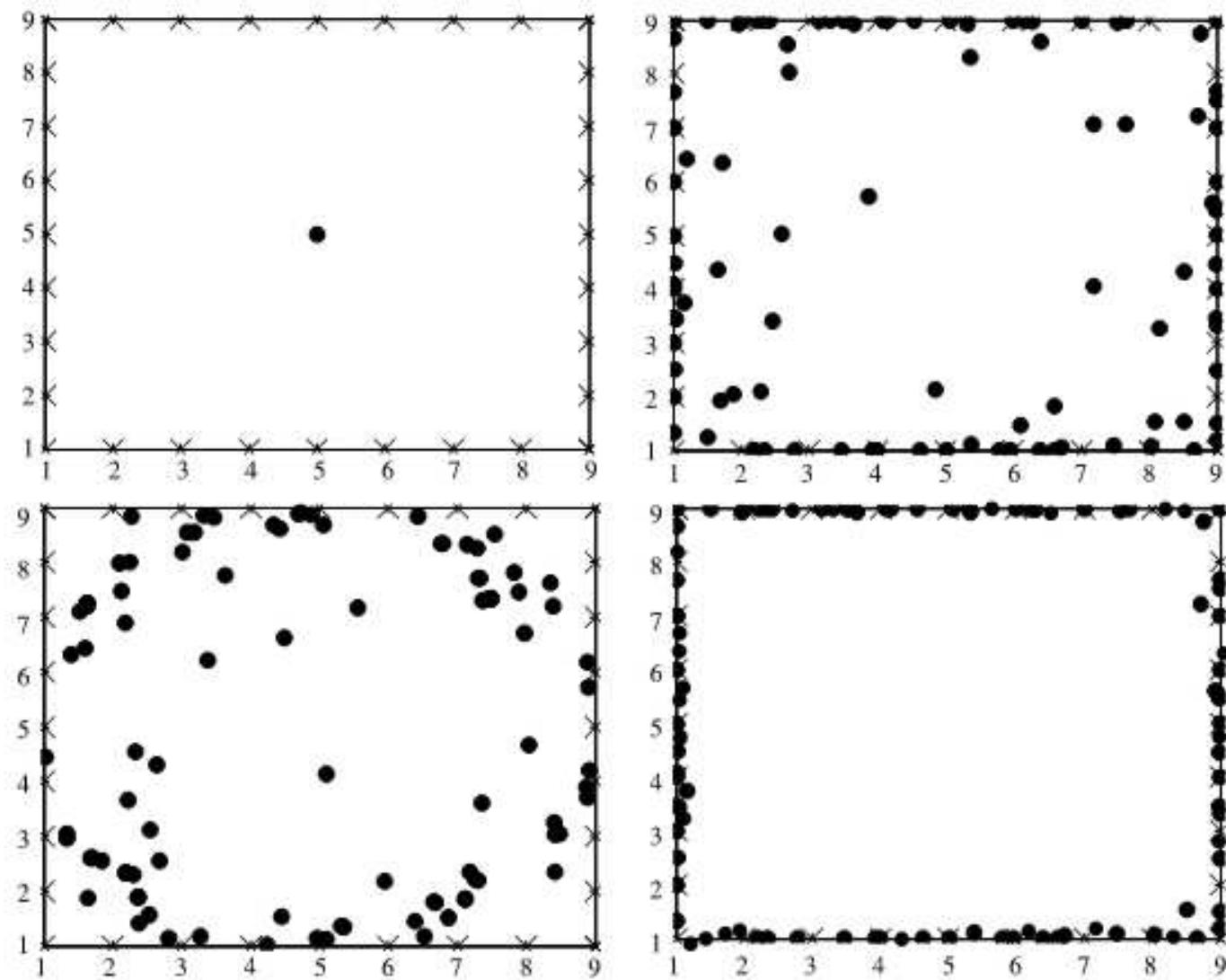
Rys. 6.38. Przemieszczanie się wag neuronów podczas procesu uczenia sieci

Analogicznie zaplanowano eksperyment w przypadku kwadratu. Na rysunku 6.39 można zaobserwować zmiany wartości wag, jakie zachodziły w procesie uczenia co 40 epok.

Gdy początkowe wartości wag sieci są wybierane (inicjalizowane) losowo, może się zdarzyć, że niektóre neurony podczas rywalizacji nigdy nie będą wygrywały. Odległość ich wag od próbek uczących oraz pozostałych neuronów staje się zbyt duża, dlatego też wagi te nie będą podlegały adaptacji. Takie jednostki nazywa się *neuronami martwymi*. Aby zwiększyć ich szanse w rywalizacji, wprowadza się pojęcie potencjału p_i w sposób następujący:

$$p_i(t+1) = \begin{cases} p_i(t) + \frac{1}{N} & \text{dla } i \neq j, \\ p_i(t) - p_{\min} & \text{dla } i = j, \end{cases} \quad (6.187)$$

gdzie j oznacza numer neuronu-zwycięzcy. Jeżeli wartość potencjału spadnie dla i -tego neuronu poniżej p_{\min} , to neuron ten nie bierze udziału we współzawodnictwie i zwycięzcy poszukuje się wśród pozostałych neuronów mających potencjał $p_i \geq p_{\min}$. Przyjmuje się, że maksymalna wartość potencjału może wynosić 1. Przy $p_{\min} = 0$ wszystkie neurony biorą udział w rywalizacji. Przy $p_{\min} = 1$ we współzawodnictwie bierze udział tylko jeden neuron, gdyż ma on potencjał umożliwiający mu odniesienie zwycięstwa.



Rys. 6.39. Ilustracja do przykładu 6.9

6.5.2. Sieci typu WTM

Dotychczas rozważaliśmy sieci samoorganizujące, w których w procesie uczenia tylko jeden neuron, tzw. zwycięzca, mógł modyfikować swoje wag. W tym punkcie zajmiemy się algorytmami, w wyniku działania których oprócz zwycięzcy korekty wag dokonują także neurony z jego sąsiedztwa. Algorytmy takie noszą nazwę *algorytmów WTM* (ang. *Winner Takes Most*). Pierwszym z nich będzie *algorytm gazu neuronowego*. Korekta wag w tym przypadku jest bardzo zbliżona do metody WTA. Najpierw obliczamy odległości wektora wejściowego od wektorów wszystkich wag w sieci. W przypadku metody WTA modyfikowaliśmy wagi neuronu, który znajdował się najbliżej podanego wejścia zgodnie z zależnością (6.184). W algorytmie gazu neuronowego sortujemy rosnąco wszystkie wektory wag neuronów w zależności od ich odległości od sygnału wejściowego. Im odległość ta będzie mniejsza, tym większa będzie zmiana wartości wag neuronu. Niech $m(i)$ oznacza kolejność i -tego neuronu uzyskaną w wyniku sortowania, przy czym dla neuronu j (zwycięzcy) mamy $m(j) = 0$, a dla neuronu najbardziej odległego $m(i) = N - 1$. Wprowadźmy zatem funkcję sąsiedztwa dla i -tego neuronu w następujący sposób:

$$G(i, \mathbf{x}) = \exp\left(-\frac{m(i)}{R}\right), \quad (6.188)$$

gdzie R jest promieniem sąsiedztwa. Wagi neuronów modyfikuje się zgodnie z zależnością

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta G(i, \mathbf{x})[\mathbf{x} - \mathbf{w}_i], \quad (6.189)$$

w której \mathbf{w}_i oznacza wektor wag w i -tym neuronie, t — numer kroku iteracji, η — współczynnik uczenia. We wzorze (6.188), w przypadku gdy $R \rightarrow 0$, tylko zwycięzca modyfikuje swoje wagi (otrzymujemy wtedy algorytm WTA). Natomiast dla $R > 0$ wagi wszystkich neuronów są aktualizowane, lecz współczynnik uczenia odległych neuronów szybko dąży do zera. Aby uzyskać dobre rezultaty, należy na początku algorytmu przyjąć dużą wartość R i następnie zmniejszać ją wraz ze wzrostem liczby jego iteracji.

Kolejnym algorymem typu WTM jest klasyczny *algorytm Kohonena*. W tej metodzie wprowadza się sąsiedztwo na stałe, tzn. definiuje się, które neurony łączą się ze sobą, tworząc siatkę. Na rysunku 6.40 przedstawiamy kilka ich przykładów, które można zdefiniować w środowisku Matlab.

W algorytmie tym oblicza się odległości wszystkich neuronów od sygnału wejściowego. Następnie wyłania się zwycięzcę (najmniejsza odległość) i modyfikuje się jego wagi oraz wagi jego sąsiadów zgodnie z zależnością (6.189), gdzie

$$G(i, \mathbf{x}) = \begin{cases} 1 & \text{dla } d(i, j) \leq R, \\ 0 & \text{dla pozostałych.} \end{cases} \quad (6.190)$$

W zależności (6.190) $d(i, j)$ oznacza odległość euklidesową między neuronem j (zwycięzca) i i -tym neuronem z sąsiedztwa G lub też odległość mierzoną liczbą neuronów. Promień sąsiedztwa R powinien maleć wraz z czasem uczenia. Często uzależnia się wielkość korekty od zwycięzcy, a także od wielkości promienia R , tzn.

$$G(i, \mathbf{x}) = \exp\left(-\frac{d^2(i, j)}{2R^2}\right), \quad (6.191)$$

gdzie j jest numerem neuronu zwycięzcy. W takiej sytuacji poszczególne neurony w różnym stopniu podlegają adaptacji. Jak można zauważyć, wraz ze wzrostem odległości od zwycięzcy wartość funkcji $G(i, \mathbf{x})$ maleje, a więc maleje też wielkość korekty wag. Sąsiedztwo określone wzorem (6.190) nosi nazwę *sąsiedztwa typu prostokątnego*, a sąsiedztwo (6.191) *typu gaussowskiego*. Duży wpływ na proces uczenia ma wielkość parametru η . W literaturze można znaleźć następujące strategie jego doboru:

a) Liniowe zmniejszanie współczynnika uczenia

$$\eta(t) = \frac{\eta_0}{T}(T-t), \quad t = 1, 2, \dots,$$

gdzie T oznacza maksymalną liczbę iteracji algorytmu uczenia, natomiast η_0 jest początkowym współczynnikiem uczenia.

b) Wykładnicze zmniejszanie współczynnika uczenia

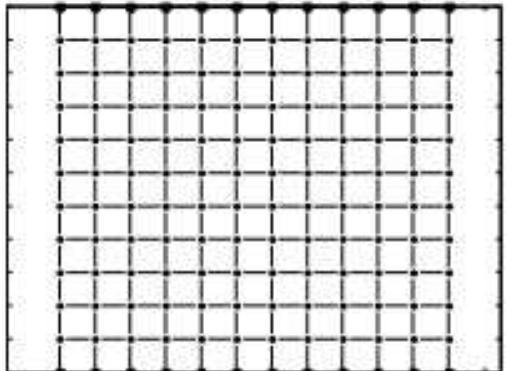
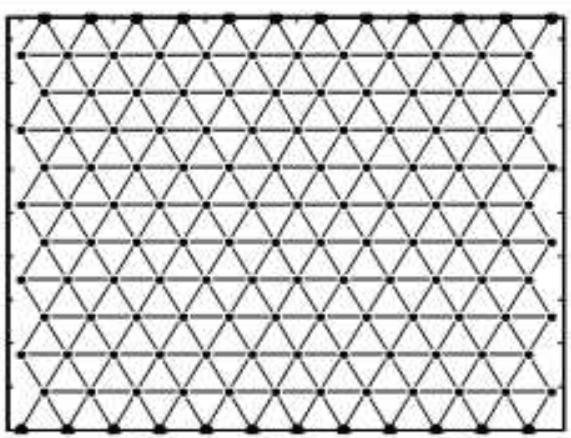
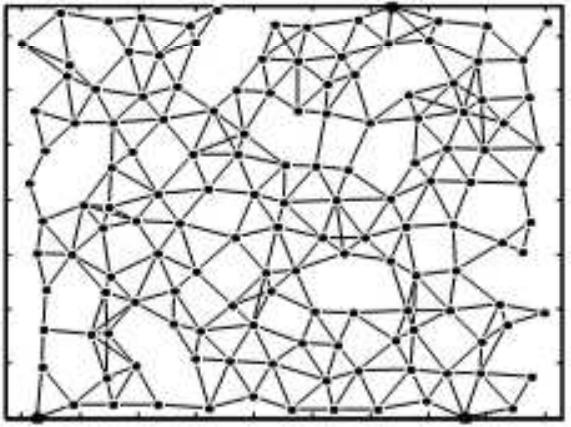
$$\eta(t) = \eta_0 e^{-Ct}, \quad t = 1, 2, \dots,$$

gdzie $C > 0$ jest pewną stałą.

c) Hiperboliczne zmniejszanie współczynnika uczenia

$$\eta_0 = \frac{C_1}{C_2 + t}, \quad t = 1, 2, \dots,$$

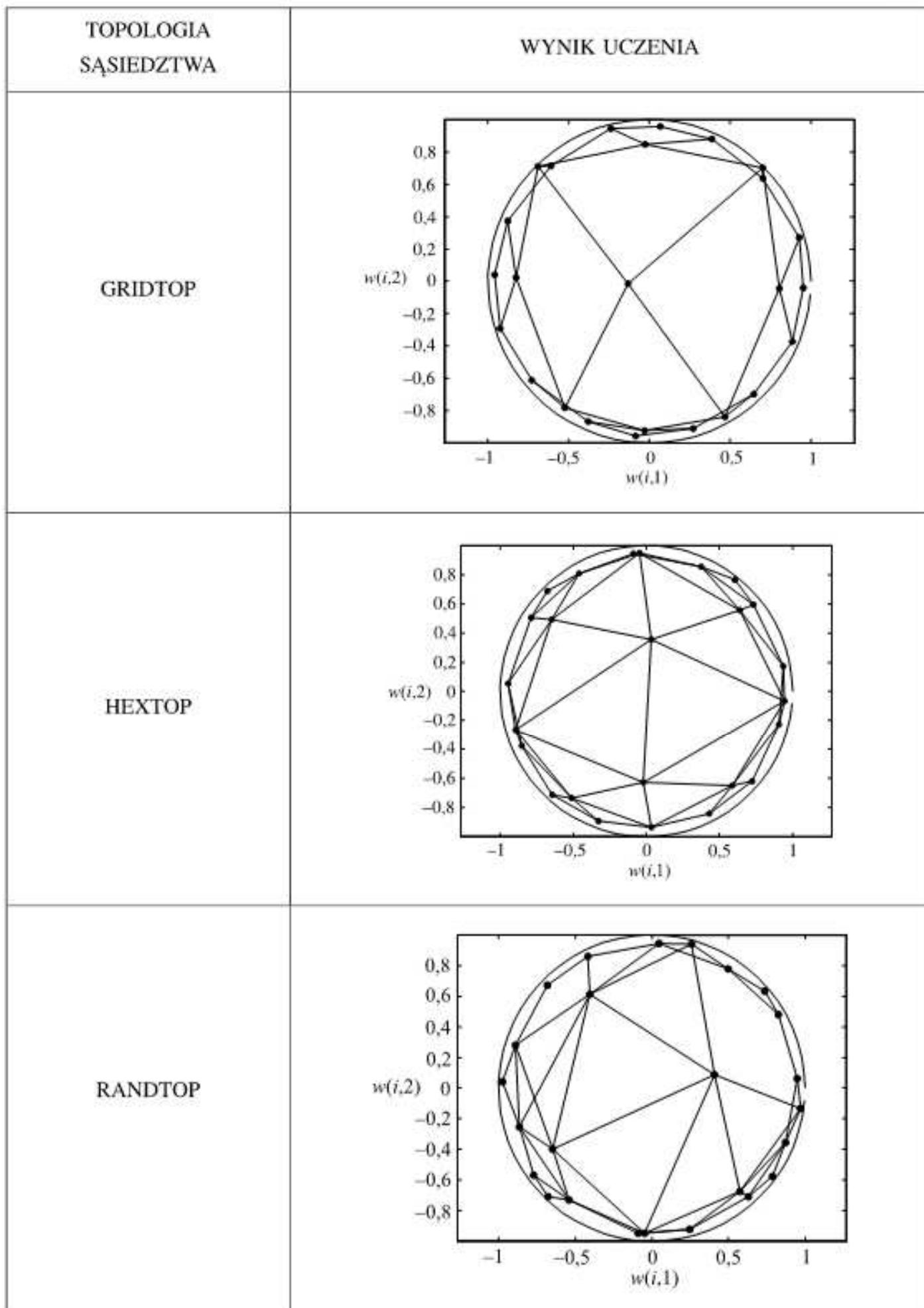
gdzie $C_1, C_2 > 0$ są pewnymi stałymi.

NAZWA SIATKI I JEJ ROZMIAR	SCHEMAT
GRIDTOP 12×12	
HEXTOP 12×12	
RANDTOP 12×12	

Rys. 6.40. Przykłady różnych topologii sąsiedztwa (kropki oznaczają położenie neuronów)

Przykład 6.10

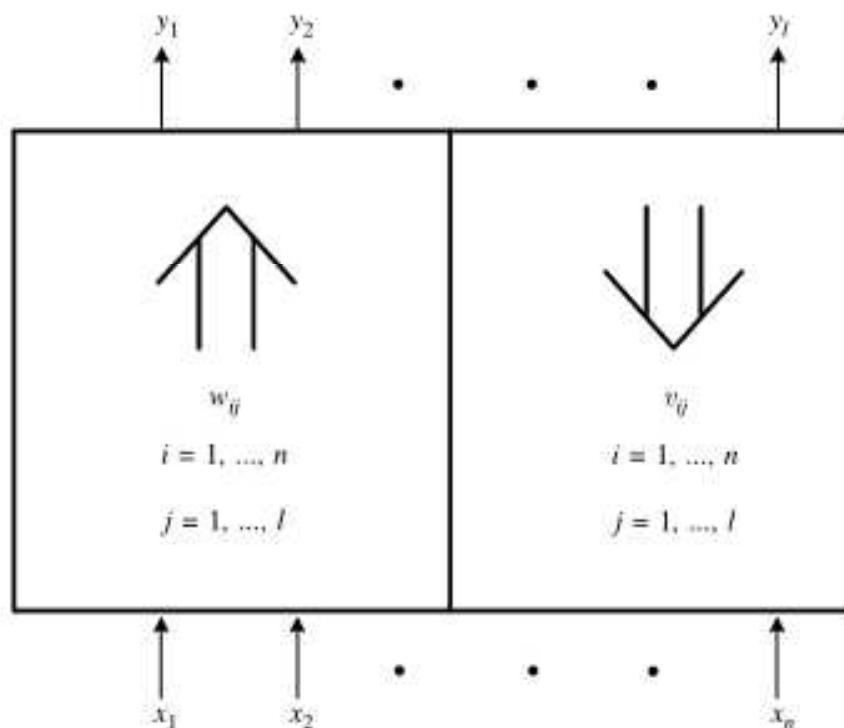
Przedstawimy teraz przykład zastosowania sieci Kohonena do rozpoznawania okręgu przy różnych topologiach sieci. Na płaszczyźnie zdefiniowaliśmy okrąg. Naszym zadaniem jest taka modyfikacja wag, które są na starcie algorytmu położone w jego środku, aby po zakończeniu procesu uczenia znalazły się one na jego brzegu. Rysunek 6.41 przedstawia położenie wag neuronów po procesie uczenia dla różnych topologii sieci, przy czym $n = 2$, $N = 25$.



Rys. 6.41. Rozkład wag neuronów w przykładzie 6.10

6.6. Sieci typu ART

Adaptacyjna teoria rezonansu (ang. *Adaptive Resonance Theory*), w skrócie ART, dotyczy uczenia sieci neuronowych bez nauczyciela. Celem działania sieci jest rozpoznanie konkurencyjne zapamiętanych wcześniej obrazów binarnych. Ponadto sieć ta powinna charakteryzować się możliwością nauczenia jej rozróżniania kształtów dotychczas nieznanych, jeżeli zachodzi taka potrzeba. Nowe wzorce są zapamiętywane w sieci wówczas, gdy stopień podobieństwa do aktualnie przechowywanych obrazów jest zbyt mały. Określa go tzw. próg czujności sieci, τ . Istnieje odpowiednik sieci dla obrazów ciągłych, tzw. ART2 zmodyfikowany później do ART3, obydwa nadające się zresztą również do klasyfikacji obrazów binarnych. Sieć ART składa się z dwóch warstw. Warstwa pierwsza, wejściowa, składa się z n neuronów. Jej zadanie polega na porównywaniu obrazów wejściowych z przechowywanymi w pamięci w celu określenia stopnia podobieństwa. Nazywana jest też warstwą porównującą. Warstwa druga, wyjściowa, składa się z m neuronów i ma za zadanie rozpoznać kształt wejściowy, to znaczy określić klasę, do której przynależy obraz. Obie warstwy oddziałują wzajemnie na siebie i wypracowują ostateczną decyzję o rozpoznaniu kształtu lub nauczeniu nowego.



Rys. 6.42. Przepływ sygnałów w sieci ART1

Na rysunku 6.42 przedstawiamy schematycznie strukturę połączeniową sieci ART. Oznaczmy wektor sygnału (kształtu) wejściowego przez $\mathbf{x} = [x_1, \dots, x_n]^T$ oraz wektor sygnałów wyjściowych górnej warstwy przez $\mathbf{y} = [y_1, \dots, y_l]^T$. Wagi połączeń pomiędzy wyjściami neuronów górnej warstwy a wejściami neuronów dolnej warstwy oznaczamy przez v_{ij} , $i = 1, \dots, n$, $j = 1, \dots, l$. Wagi połączeń pomiędzy wyjściami neuronów dolnej warstwy a wejściami neuronów górnej warstwy oznaczamy przez w_{ij} . Charakterystyczne jest zatem istnienie dwóch rodzajów połączeń międzyneuronowych obu warstw, „z dołu do góry” (wagi w_{ij}) i „z góry na dół” (wagi v_{ij}). Przedstawimy

teraz algorytm uczenia sieci ART1. W algorytmie tym można wyróżnić następujące kroki:

Krok 1:

Działanie algorytmu rozpoczynamy od inicjalizacji wartości wag w obu zestawach połączeń. Zazwyczaj dla początkowych wartości wag połączeń z góry na dół mamy $v_{ij}(0) = 1$, a połączeń z dołu do góry przyjmujemy $w_{ij}(0) = \frac{1}{1+n}$, gdzie n jest liczbą neuronów warstwy wejściowej. Jako próg czujności τ przyjmuje się małą liczbę dodatnią z przedziału $(0, 1)$.

Krok 2:

W kroku tym po podaniu wektora wejściowego $\mathbf{x} = [x_1, \dots, x_n]^T$, gdzie $x_i \in \{0, 1\}$, $i = 1, 2, \dots, n$, następuje wyliczenie sumy $\sum_{i=1}^n w_{ij} x_i$, $j = 1, \dots, l$. Suma ta jest traktowana jako miara dopasowania obrazu do aktualnie pamiętanych wzorców. Maksymalna jej wartość decyduje o tym, który neuron warstwy wyjściowej (oznaczmy jego numer przez j^*) przyjmie wartość 1, czyli zostanie zwycięzcą rywalizacji. Jednocześnie wskazuje on na klasę, do której kwalifikuje rozpoznawany kształt. Pozostałe neurony na swych wyjściach przyjmują wartość 0. Jest to normalna faza pracy sieci rozpoznającej kształt uprzednio zapamiętany.

Krok 3:

Po wskazaniu przez neuron-zwycięzcę klasy, do której badany obraz najlepiej pasuje, konieczne jest określenie, jak bardzo jest on podobny do wektorów wcześniej przypisanych do tej klasy. Następuje sprawdzenie, czy

$$D = \frac{\sum_{i=1}^n v_{ij^*} x_i}{\sum_{i=1}^n x_i} > \tau. \quad (6.192)$$

Jeśli nie, to następuje wyzerowanie wyjścia aktualnego neuronu-zwycięzcy, po czym za neuron zwycięski przyjmuje się najbardziej pobudzony spośród pozostałych neuronów. Następnie przechodzimy do punktu 3. Jeśli w wyniku tej operacji (która może być wielokrotnie powtarzana) nie uda się osiągnąć pozytywnego rezultatu, to należy przyporządkować badany kształt do nowej, nieznanej dotąd klasy, przypisując jej pierwszy, wolny dotąd neuron warstwy wyjściowej. Jeśli test stopnia podobieństwa wypadnie pozytywnie, mówimy, że sieć i sygnał wejściowy są w „rezonansie”, co uzasadnia nazwę sieci ART. Wybór malej wartości progu czujności τ (tzn. bliskiego zera) powoduje, że spełnienie warunku (6.192) jest możliwe nawet przy stosunkowo malej liczbie zgodnych pikseli wzorca i rozpoznawanego obrazu. Sieć klasyfikuje wówczas obrazy, czasem dość mało podobne, do niewielkiej liczby możliwych klas. Można powiedzieć, że sieć klasyfikuje wówczas obrazy „z grubsza”. Gdy parametr czujności zbliża się do jedynki, wówczas tworzone są liczne kategorie z dość subtelnymi różnicami pomiędzy nimi i zachodzi dość precyzyjne rozpoznawanie obrazów przez sieć.

Krok 4:

Po wytypowaniu neuronu-zwycięzcy następuje korekcja związanych z nim wag, zarówno wejściowych w_{ij^*} , jak i v_{ij^*} zgodnie ze wzorami

$$v_{ij^*}(t+1) = v_{ij^*}(t) x_i, \quad (6.193)$$

$$w_{ij*}(t+1) = \frac{v_{ij*}(t+1)}{0,5 + \sum_{j=1}^n v_{ij*}(t+1)x_i}. \quad (6.194)$$

Zwróćmy uwagę na fakt, że w tym kroku następuje „douczenie” sieci, które jest rozumiane jako adaptacja przechowywanego wzorca do aktualnie rozpoznanego kształtu. Jest to osiągane przez swoiste nałożenie na siebie obu kształtów i ma na celu osiągnięcie ich binarnej zgodności. Wagi połączeń dla zgodnych bitów są wzmacniane, a dla niezgodnych — osłabiane lub pozostawiane bez zmian.

Krok 5:

Przetwarzanie sygnału wejściowego przez sieć ART zostaje przerwane, gdy wartości wag nie ulegają zmianie lub też dla żadnego z neuronów nie jest spełniony warunek (6.192) i w warstwie pierwszej nie istnieje żaden „wolny” neuron, do którego moglibyśmy przypisać nową klasę.

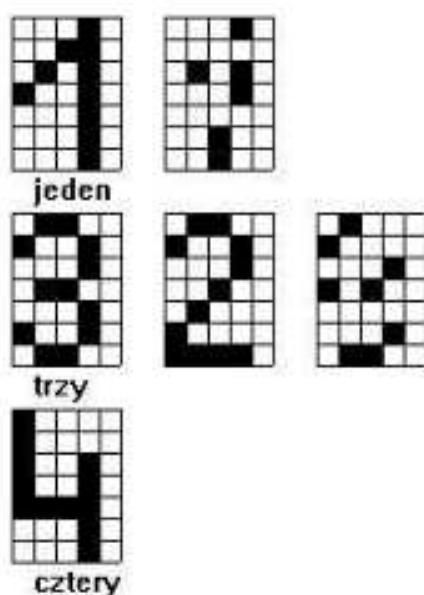
Sieć ART posługuje się dwoma zbiorami wag, które w istocie stanowią o pamięci sieci, czyli jej zdolności rozpoznawania. Zestaw wag „z dołu do góry” odpowiada za tzw. *pamięć długotrwałą*, a zestaw wag „z góry na dół” za tzw. *pamięć krótkotrwałą*. Trzeba zauważyć, że ze względu na sposób douczania się (modyfikacje wag w kroku 4 algorytmu) i stosunkowo nieskomplikowany test podobieństwa (6.192) sieć nie radzi sobie dobrze z rozpoznawaniem obrazów zniekształconych przez obecność zakłóceń, zaszumionych. W tych warunkach możliwe jest jedynie rozpoznawanie „z grubsza”, czyli wybór małego parametru czujności τ .

Przykład 6.11

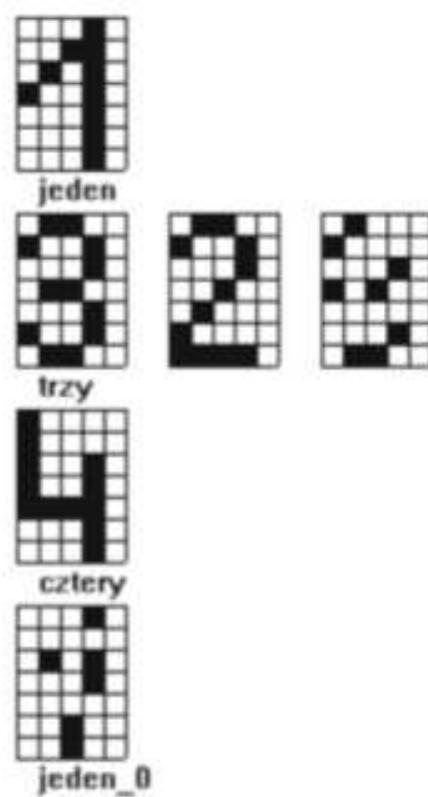
W celu ilustracji działania sieci typu ART1 rozważmy zagadnienie klasyfikacji kolejnych cyfr od 1 do 4. Ich binarną reprezentację podano na wejście sieci dla różnych wartości progu czujności. Na rysunku 6.43 przedstawiono wyniki symulacji, w której sekwencja podawanych na wejście kształtów była następująca: 1, 3, 2, 4, zniekształcona cyfra 1 i zniekształcona cyfra 3. Parametr czujności był równy 0,6, czyli powiedzmy „średni”. Jak widać, sieć nie rozróżniła kształtów 3 i 2 i sklasyfikowała je jako jednakowe (utożsamione ze wzorcem 3). Natomiast zniekształcona cyfra 1 została prawidłowo zaklasyfikowana do klasy wzorca 1, a zniekształcona cyfra 3 również właściwie, tj. do klasy wzorca 3.

Rysunek 6.44 ilustruje sytuację, gdy próg czujności sieci został zwiększy do $\tau = 0,75$. Sekwencja podawanych obrazów była taka jak poprzednio. W tym przypadku zniekształcona cyfra 1 została uznana za nowy (czwarty) wzorzec. Rysunek 6.45 przedstawia wynik symulacji, gdy próg czujności wynosi 0,4 (jest „mały”). Wówczas cyfry 1 i 4 należą do tej samej klasy (to znaczy, że są nieroóżniane przez sieć), wzorce 2, 3 oraz zniekształcona cyfra 3, podobnie jak poprzednio, zostały zakwalifikowane do klasy drugiej, natomiast zniekształcona cyfra 1 jest wzorcem nowej klasy. Rysunek 6.46 pokazuje przypadek dla progu czujności równego 0,9 (bardzo duży). Obecnie każdy kształt badanej sekwencji jest wzorcem odrębnej klasy, ale podanie zniekształconej cyfry 3 powoduje komunikat o niemożności utworzenia nowej klasy, czyli świadczy o tym, że kształt ten „nie pasuje” do żadnego z wcześniejszych wzorców.

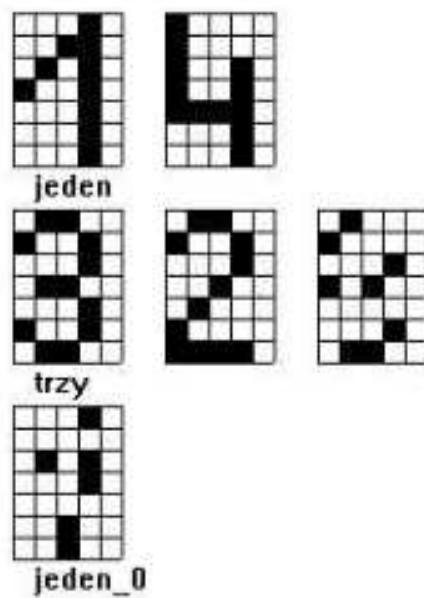
Powyższy przykład jest jedynie ilustracją właściwości sieci ART1, pokazującym jej bardzo interesujące możliwości rozpoznawania kształtów, uczenia się bez nadzoru,



Rys. 6.43. Wynik działania sieci ART1 — próg czuwności $\tau = 0,6$



Rys. 6.44. Wynik działania sieci ART1 — próg czuwności $\tau = 0,75$



Rys. 6.45. Wynik działania sieci ART1 — próg czuwności $\tau = 0,4$



Rys. 6.46. Wynik działania sieci ART1 — próg czuwności $\tau = 0,9$

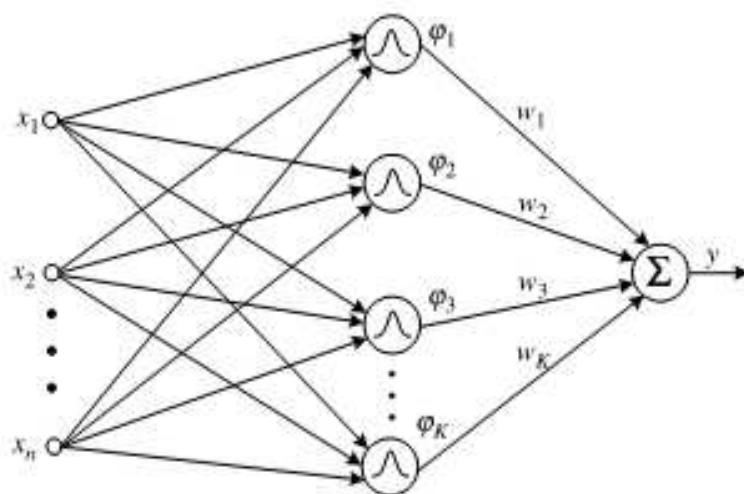
tj. zdolność samodzielnego podejmowania decyzji o tworzeniu nowej, nieznanej klasy. Jednocześnie widać, jak trudną sprawą jest właściwy dobór progu czuwności sieci, który, jak można przypuszczać, powinien być ustalany przez eksperta dla konkretnego zadania.

6.7. Sieci radialne

Sieci radialne składają się z neuronów, których funkcje aktywacji realizują odwzorowanie

$$\mathbf{x} \rightarrow \varphi(\|\mathbf{x} - \mathbf{c}\|), \quad \mathbf{x} \in \mathbf{R}^n, \quad (6.195)$$

gdzie ($\|\cdot\|$) najczęściej oznacza normę euklidesową. Funkcje $\varphi(\|\mathbf{x} - \mathbf{c}\|)$ nazywamy *radialnymi funkcjami bazowymi*. Ich wartości zmieniają się radialnie wokół środka \mathbf{c} . Na rysunku 6.47 przedstawiono schemat sieci neuronowej radialnej nazywanej *siecią RBF* (ang. *Radial Basis Functions*). Sygnały wejściowe x_1, x_2, \dots, x_n tworzące wektor \mathbf{x} podawane są na każdy z neuronów w warstwie ukrytej (tożsamej z warstwą wejściową).



Rys. 6.47. Schemat sieci radialnej

Neurony w warstwie ukrytej realizują odwzorowanie (6.195). Liczba tych neuronów jest równa liczbie wektorów \mathbf{x} w zbiorze uczącym lub mniejsza. Neuron w warstwie wyjściowej realizuje operację ważonej sumy sygnałów wyjściowych neuronów w warstwie ukrytej, co można wyrazić za pomocą następującego wzoru:

$$y = \sum_i w_i \varphi_i = \sum_i w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|). \quad (6.196)$$

Poniżej przedstawiamy typowe przykłady radialnych funkcji bazowych:

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = \exp(-\|\mathbf{x} - \mathbf{c}_i\|^2/r^2). \quad (6.197)$$

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = \|\mathbf{x} - \mathbf{c}_i\|/r^2, \quad (6.198)$$

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = (r^2 + \|\mathbf{x} - \mathbf{c}_i\|^2)^{-\alpha}, \quad \alpha > 0, \quad (6.199)$$

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = (r^2 + \|\mathbf{x} - \mathbf{c}_i\|^2)^\beta, \quad 0 < \beta < 1, \quad (6.200)$$

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = (r \|\mathbf{x} - \mathbf{c}_i\|)^2 \ln(r \|\mathbf{x} - \mathbf{c}_i\|), \quad (6.201)$$

przy czym $r > 0$. W celu uproszczenia zapisu chwilowo przyjmujemy, że parametry skalarne we wzorach (6.197)–(6.201) są takie same dla wszystkich neuronów w sieci. Analiza działania sieci radialnej obejmuje dwa przypadki w zależności od liczby neuronów w warstwie ukrytej:

Przypadek a

Założymy, że na wejście sieci rozwiązującej problem interpolacji podano M różnych wektorów wejściowych $\mathbf{x}(1), \dots, \mathbf{x}(M)$, które mają być odwzorowane w zbiór liczb

rzeczywistych $d(1), \dots, d(M)$. Problem polega na znalezieniu funkcji F realizującej odwzorowanie $\mathbf{R}^n \rightarrow \mathbf{R}$, tak aby zachodziła równość

$$F(\mathbf{x}_i) = d_i \quad (6.202)$$

dla $i = 1, 2, \dots, M$. Jako funkcję F przyjmiemy

$$F(\mathbf{x}) = \sum_{i=1}^M w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|). \quad (6.203)$$

Założymy, że centra $\mathbf{c}_1, \dots, \mathbf{c}_M$ są równe kolejnym wartościom wektorów $\mathbf{x}(1), \dots, \mathbf{x}(M)$. Podstawiając warunek (6.202) do zależności (6.203), otrzymujemy następujące równanie macierzowe:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \dots & \varphi_{1M} \\ \varphi_{21} & \varphi_{22} & \dots & \varphi_{2M} \\ \dots & \dots & \dots & \dots \\ \varphi_{M1} & \varphi_{M2} & \dots & \varphi_{MM} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_M \end{bmatrix}, \quad (6.204)$$

gdzie $\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|)$. Przy oznaczeniach

$$\mathbf{d} = [d_1, d_2, \dots, d_M]^T, \quad (6.205)$$

$$\mathbf{w} = [w_1, w_2, \dots, w_M]^T, \quad (6.206)$$

$$\Phi = \{\varphi_{ji} \mid j, i = 1, 2, \dots, M\} \quad (6.207)$$

zależność (6.204) przybiera postać

$$\Phi \mathbf{w} = \mathbf{d}. \quad (6.208)$$

Dla pewnej klasy radialnych funkcji bazowych $\varphi(\|\mathbf{x} - \mathbf{c}\|)$, np. dla funkcji (6.197) i (6.199), macierz Φ jest dodatnio określona, jeżeli wektory wejściowe spełniają warunek

$$\mathbf{x}(1) \neq \mathbf{x}(2) \neq \dots \neq \mathbf{x}(M). \quad (6.209)$$

Wówczas rozwiązanie równania (6.208) możemy zapisać następująco:

$$\mathbf{w} = \Phi^{-1} \mathbf{d}. \quad (6.210)$$

Przypadek b

Przyjęcie liczby neuronów radialnych równej liczbie wzorców uczących powoduje, że sieć traci możliwość uogólniania. Ponadto przy dużej liczbie sygnałów wzorcowych struktura sieci musiałaby rozrosnąć się do ogromnych rozmiarów. Dlatego dąży się do tego, aby struktura radialna realizowała odwzorowanie

$$F(\mathbf{x}) = \sum_{i=1}^K w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|), \quad (6.211)$$

gdzie $K < M$. W ten sposób poszukujemy rozwiązania przyblizonego. Musimy więc starać się odpowiednio dobrze nie tylko wag w_i , ale także znaleźć środki \mathbf{c}_i dla neuronów radialnych. Odpowiednie kryterium minimalizacji przybiera postać

$$E = \sum_{i=1}^M \left[\sum_{j=1}^K w_j \varphi(\|\mathbf{x}_i - \mathbf{c}_j\|) - d_i \right]^2. \quad (6.212)$$

Przyjmijmy następujące oznaczenia:

$$\mathbf{G} = \begin{bmatrix} \varphi(\|\mathbf{x}_1 - \mathbf{c}_1\|) & \varphi(\|\mathbf{x}_1 - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{x}_1 - \mathbf{c}_K\|) \\ \varphi(\|\mathbf{x}_2 - \mathbf{c}_1\|) & \varphi(\|\mathbf{x}_2 - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{x}_2 - \mathbf{c}_K\|) \\ \dots & \dots & \dots & \dots \\ \varphi(\|\mathbf{x}_M - \mathbf{c}_1\|) & \varphi(\|\mathbf{x}_M - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{x}_M - \mathbf{c}_K\|) \end{bmatrix}, \quad (6.213)$$

$$\mathbf{d} = [d_1, d_2, \dots, d_M]^T, \quad (6.214)$$

$$\mathbf{w} = [w_1, w_2, \dots, w_M]^T. \quad (6.215)$$

Macierz \mathbf{G} zdefiniowana zależnością (6.213) jest tzw. *macierzą Greena*. Minimalizując kryterium (6.212) oraz uwzględniając wzory (6.213)–(6.215), otrzymujemy

$$\mathbf{Gw} = \mathbf{d}, \quad (6.216)$$

$$\mathbf{w} = \mathbf{G}^+ \mathbf{d}, \quad (6.217)$$

gdzie \mathbf{G}^+ oznacza pseudoinwersję macierzy prostokątnej \mathbf{G} , tzn.

$$\mathbf{G}^+ = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T. \quad (6.218)$$

Do kryterium (6.212) można dodać tzw. *czynnik regularizacyjny*, który wynika z metody regularizacji Tichonowa

$$E(f) = \sum_{i=1}^M \left[\sum_{j=1}^K w_j \varphi(\|\mathbf{x}_i - \mathbf{c}_j\|)^2 - d_i \right] + \lambda \|\mathbf{P}f\|^2 = \|\mathbf{Gw} - \mathbf{d}\|^2 + \lambda \|\mathbf{P}f\|^2, \quad (6.219)$$

gdzie \mathbf{P} jest pewnym liniowym operatorem różniczkowym. W zadaniach aproksymacji wprowadzenie tego operatora wiąże się z założeniem o gładkości funkcji, która ma przybliżać nieznane rozwiązanie. Prawą stronę wyrażenia (6.219) można przedstawić w postaci [73]

$$\|\mathbf{P}f\|^2 = \mathbf{w}^T \mathbf{G}_0 \mathbf{w}, \quad (6.220)$$

gdzie macierz \mathbf{G}_0 jest macierzą kwadratową $K \times K$, określona następująco:

$$\mathbf{G}_0 = \begin{bmatrix} \varphi(\|\mathbf{c}_1 - \mathbf{c}_1\|) & \varphi(\|\mathbf{c}_1 - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{c}_1 - \mathbf{c}_K\|) \\ \varphi(\|\mathbf{c}_2 - \mathbf{c}_1\|) & \varphi(\|\mathbf{c}_2 - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{c}_2 - \mathbf{c}_K\|) \\ \dots & \dots & \dots & \dots \\ \varphi(\|\mathbf{c}_K - \mathbf{c}_1\|) & \varphi(\|\mathbf{c}_K - \mathbf{c}_2\|) & \dots & \varphi(\|\mathbf{c}_K - \mathbf{c}_K\|) \end{bmatrix}. \quad (6.221)$$

Minimalizując zależność (6.219) oraz wykorzystując równania (6.220) i (6.221), otrzymujemy poszukiwany przez nas wektor wag \mathbf{w}

$$\mathbf{w} = (\mathbf{G}^T \mathbf{G} + \lambda \mathbf{G}_0)^{-1} \mathbf{G}^T \mathbf{d}. \quad (6.222)$$

W literaturze wykazano, że sieć neuronowa o bazowych funkcjach radialnych opartych na funkcjach Greena jest uniwersalnym aproksymatorem.

Algorytm uczenia sieci radialnej składa się z dwóch etapów:

1. Najpierw dobiera się położenie oraz kształt funkcji bazowych wybranymi metodami:

- dobór losowy,
- dobór przy zastosowaniu procesu samoorganizacji,
- dobór metodą wstępnej propagacji błędów.

2. W drugiej fazie dobierana jest macierz wag warstwy wyjściowej, przy czym problem ten jest znacznie łatwiejszy do rozwiązania niż dobór parametrów funkcji bazowych. Macierz wag w wyznacza się w jednym kroku przez pseudoinwersję macierzy Greena G , tzn. $w = G^+d$.

Przedstawimy teraz jedną z wymienionych metod doboru parametrów z wykorzystaniem procesu samoorganizacji. Proces samoorganizacji dzieli przestrzeń sygnałów wejściowych na tzw. *obszary Voronoia*. Każdy taki obszar nazywamy *grupą* lub *klasterem*. Zawiera on punkt centralny, który jest średnią wszystkich elementów znajdujących się w grupie. Jest to jednocześnie centrum funkcji radialnej. Istnieją dwie wersje algorytmu:

1. bezpośrednia — aktualizacja centrów następuje po każdej prezentacji wektora x z ciągu uczącego;

2. skumulowana — aktualizacja następuje po zaprezentowaniu wszystkich wektorów uczących.

Stosuje się także odmianę wersji bezpośredniej tego algorytmu, w której adaptacji podlegają również centra z najbliższego sąsiedztwa zwycięzcy. Są one modyfikowane zgodnie z regułą WTM.

Inną metodę dobierania parametrów dla sieci RBF stosuje się w programie Matlab. W środowisku tym istnieją dwie metody tworzące taką strukturę. W przypadku pierwszej z nich, poprzez instrukcję NEWRBE, dla każdego wektora uczącego $x(1), \dots, x(M)$ tworzymy oddzielny neuron radialny w pierwszej warstwie. Następnie dobierane są wagi dla neuronu z warstwy drugiej. Jest to sytuacja analogiczna do omówionego przez nas wcześniej przypadku a, dotyczącego analizy działania sieci radialnej. Natomiast druga metoda, w wyniku działania innej instrukcji o nazwie NEWRB, powoduje uruchomienie następującego algorytmu:

1. Stworzona zostaje sieć dwuwarstwowa bez żadnego neuronu w warstwie pierwszej.

2. Na wejścia sieci zostają podawane kolejne wektory uczące i dla nich zostaje obliczony błąd wyjściowy (dla każdej próbki uczącej zapamiętany zostaje błąd oddziennie).

3. Zostaje wybrany wektor uczący, dla którego błąd jest największy, i do warstwy pierwszej zostaje dodany neuron radialny o wagach równych składowym tego wektora uczącego.

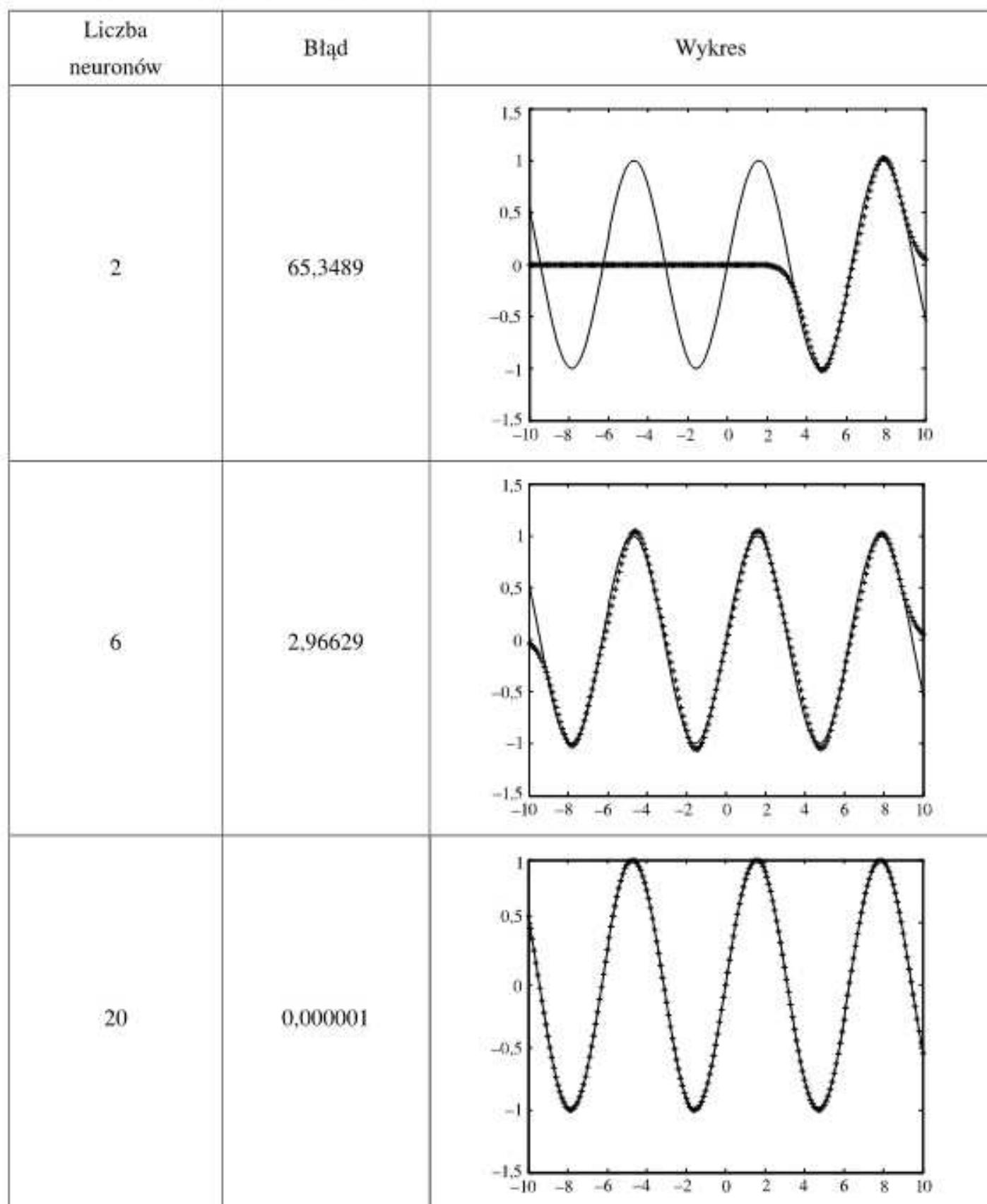
4. Następnie wagi neuronu liniowego w warstwie drugiej zostają tak dobrane, aby zminimalizować błąd sieci.

5. Punkty 2–4 powtarzane są tak długo, aż błąd działania sieci nie spadnie poniżej progu podanego przez użytkownika jako parametr funkcji NEWRB. Innym parametrem metody NEWRB jest liczba neuronów, po której przekroczeniu wykonywanie algorytmu powinno zostać przerwane. Domyślnie liczba ta jest równa rozmiarowi ciągu uczącego.

Przykład 6.12

Metodą wykorzystującą instrukcję NEWRB z pakietu Matlab wykonaliśmy doświadczenie polegające na nauczeniu sieci RBF odwzorowywania funkcji $f(x) = \sin x$, gdzie $x \in [-10, 10]$. Ciąg uczący składa się z punktów x wygenerowanych poprzez dyskrety-

zację przedziału $[-10, 10]$ z krokiem 0,1 oraz odpowiadających im wartości $f(x)$. Próg błędu, poniżej którego sieć uznajemy za nauczoną, wynosi 0,000001. Na rysunku 6.48 przedstawiamy działanie stworzonej przez nas struktury dla różnej liczby neuronów. Linia ciągła przedstawia wykres funkcji $y = \sin x$. Symbolem + oznaczyliśmy odpowiedzi sieci na podane sygnały testujące.



Rys. 6.48. Wyniki uczenia sieci RBF przy różnej liczbie neuronów radialnych

Sieci neuronowe radialne znajdują zastosowania w problemach klasyfikacyjnych, aproksymacyjnych oraz zadaniach predykcji. Są to zadania, w których od wielu lat stosowane są sieci sigmoidalne. Jednak struktury radialne stosują inny sposób przetwarzania danych, przez co następuje skrócenie procesu uczenia. Przy rozwiązywaniu zadań klasyfikacyjnych sieć radialna nie tylko dostarcza informacji, do której klasy należy wzorzec, ale także wskazuje na ewentualną możliwość utworzenia nowej klasy. Istotną zaletą jest znacznie uproszczony algorytm jej uczenia. Punkt startowy można dobrąć tak, aby znajdował się znacznie bliżej rozwiązania optymalnego niż w przypadku sieci sigmoidalnych. Podstawowe różnice pomiędzy sieciami radialnymi a sigmoidalnymi to:

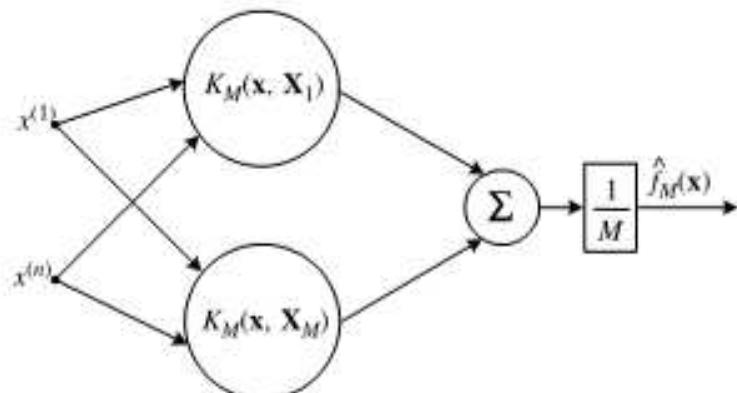
1. Sieci radialne mają z góry ustaloną architekturę składającą się z dwóch warstw, podczas gdy sieci sigmoidalne mogą mieć dowolną ich liczbę.
2. Sieci radialne w warstwie ukrytej mogą stosować różne funkcje bazowe, podczas gdy w sieci wielowarstwowej najczęściej stosuje się funkcje sigmoidalne.
3. W sieciach radialnych można stosować różne techniki uczenia w odniesieniu do obu warstw, np. warstwę pierwszą (ukrytą) można uczyć, wykorzystując metodę gradientową lub metodę samoorganizacji, natomiast warstwę drugą metodą pseudoinwersji. W przypadku sigmoidalnych sieci wielowarstwowych neurony wszystkich warstw najczęściej uczone są metodą wstecznej propagacji błędów.

6.8. Probabilistyczne sieci neuronowe

W wielu zagadnieniach identyfikacji, klasyfikacji i predykcji zachodzi potrzeba estymacji funkcji gęstości prawdopodobieństwa. W celu estymacji tej funkcji możemy zastosować estymator

$$\hat{f}_M(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M K_M(\mathbf{x}, \mathbf{X}_i), \quad (6.223)$$

gdzie $\mathbf{X}_1, \dots, \mathbf{X}_M$ jest ciągiem obserwacji n -wymiarowej zmiennej losowej \mathbf{X} o gęstości prawdopodobieństwa f , natomiast K_M jest odpowiednio dobranym jądrem.



Rys. 6.49. Probabilistyczna sieć neuronowa — estymacja funkcji gęstości

Na rysunku 6.49 przedstawiono realizację sieciową estymatora (6.223). Należy zaznaczyć, że proponowana sieć nie wymaga procesu uczenia (optymalnego doboru wag

połączeń), gdyż funkcję wag pełnią kolejne składowe wektorów obserwacji \mathbf{X}_i . Jako funkcję K_M można przyjąć tzw. *jądro Parzena* postaci

$$K_M(\mathbf{x}, \mathbf{u}) = h_M^{-n} K\left(\frac{\mathbf{x} - \mathbf{u}}{h_M}\right), \quad (6.224)$$

przy czym ciąg h_M jest funkcją długości ciągu uczącego M i powinien spełniać warunki

$$\lim_{M \rightarrow \infty} h_M = 0 \quad \text{i} \quad \lim_{M \rightarrow \infty} M h_M^n = \infty. \quad (6.225)$$

Można wykazać (Cacoullos [20]), że

$$E[\hat{f}_M(\mathbf{x}) - f_M(\mathbf{x})]^2 \xrightarrow{M} 0 \quad (6.226)$$

w punktach ciągłości gęstości f . Funkcję K we wzorze (6.224) wygodnie jest przedstawić w postaci

$$K(\mathbf{x}) = \prod_{i=1}^n H(x^{(i)}). \quad (6.227)$$

Zakładając, że funkcja H jest typu gaussowskiego, mamy

$$\hat{f}_M(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} n h_M^n} \sum_{i=1}^M \exp\left(-\frac{(\mathbf{x} - \mathbf{X}_i)^T (\mathbf{x} - \mathbf{X}_i)}{2h_M^2}\right). \quad (6.228)$$

W sposób analogiczny możemy skonstruować probabilistyczną sieć neuronową w celu estymacji funkcji regresji. Niech (\mathbf{X}, Y) będzie parą zmiennych losowych. Założymy, że \mathbf{X} przyjmuje wartości w zbiorze R^n , natomiast Y w zbiorze R . Niech f będzie funkcją gęstości prawdopodobieństwa zmiennej losowej \mathbf{X} . Na podstawie M niezależnych obserwacji $(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_M, Y_M)$ zmiennych (\mathbf{X}, Y) należy estymować funkcję regresji R zmiennej losowej Y względem \mathbf{X} , tj.

$$\phi(\mathbf{x}) = E[Y | \mathbf{X} = \mathbf{x}]. \quad (6.229)$$

Zdefiniujmy funkcję

$$R(\mathbf{x}) = \phi(\mathbf{x}) \cdot f(\mathbf{x}). \quad (6.230)$$

Do estymacji funkcji (6.230) można zastosować estymator analogiczny do procedury (6.223), tzn.

$$\hat{R}_M(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M Y_i K_M(\mathbf{x}, \mathbf{X}_i). \quad (6.231)$$

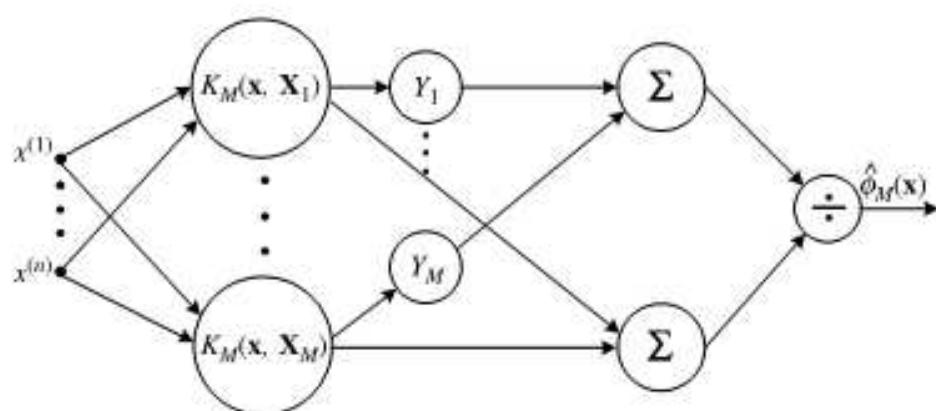
Zatem funkcję regresji estymujemy za pomocą

$$\hat{\phi}_M(\mathbf{x}) = \frac{\hat{R}_M(\mathbf{x})}{\hat{f}_M(\mathbf{x})}. \quad (6.232)$$

W konsekwencji otrzymujemy następujący estymator:

$$\hat{\phi}_M(\mathbf{x}) = \frac{\sum_{i=1}^M Y_i K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h_M}\right)}{\sum_{i=1}^M K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h_M}\right)}. \quad (6.233)$$

Na rysunku 6.50 pokazano realizację neuronową estymatora funkcji regresji nawiązującą w swojej konstrukcji do struktury danej na rysunku 6.49. Przedstawione struktury probabilistycznych sieci neuronowych nie wymagają uczenia. Ponadto zastosowanie tych sieci przy odpowiednio dobranym ciągu h_M gwarantuje zbieżność estymatorów. Są to jednak wyniki asymptotyczne. W praktyce musimy zagwarantować posiadanie ciągów uczących o znacznej długości. Obie struktury można zastosować do rozwiązania zagadnienia klasyfikacji i wówczas uzyskujemy zbieżność do reguły bayesowskiej.



Rys. 6.50. Probabilistyczna sieć neuronowa do estymacji funkcji regresji

6.9. Uwagi

Wiedza na temat biologicznego neuronu przyczyniła się do stworzenia jego matematycznego modelu. Pionierami w tej dziedzinie byli amerykańscy uczeni McCulloch i Pitts, którzy w roku 1943 stworzyli pierwszy model neuronu [133]. Ich pomysł był rozwijany aż do roku 1969, w którym Minsky i Papert opublikowali książkę [138] akcentującą ograniczone możliwości sztucznych sieci neuronowych jednowarstwowych, a jednocześnie wskazującą na brak algorytmów uczenia sieci wielowarstwowych. Książka ta spowodowała zatrzymanie prac badawczych nad sieciami neuronowymi. Do tego jednak czasu na świecie ukazało się wiele publikacji w tym zakresie. Między innymi Donald Hebb opracował regułę uczenia sieci zwaną dziś „regułą Hebba” [74], Rosenblatt zbudował perceptron [181], natomiast Widrow skonstruował model neuronu o nazwie Adaline [257–259]. Powrót do badań nad sieciami neuronowymi spowodował opublikowanie w 1986 roku pracy [183]. Opracowanie to przedstawiało opis metody uczenia wielowarstwowych sieci neuronowych, która została nazwana metodą „wstecznej propagacji błędów”. Fakt ten spowodował przywrócenie do łask sieci neuronowych. Do dziś są one przedmiotem badań dziesiątek tysięcy uczonych na całym świecie. Probabilistyczne sieci neuronowe zaproponował Specht [236, 237]. Koncepcja ich wywodzi się z nieparametrycznych metod estymacji funkcji gęstości i regresji [20, 69, 157, 159, 188–203]. Probabilistyczne sieci neuronowe mają również zastosowanie w przypadku niestacjonarnych rozkładów prawdopodobieństw [221–224]. Algorytm RLS uczenia sieci neuronowych podano w pracy [11]. Spośród wielu monografii i podręczników na temat sieci neuronowych można wymienić [12, 26, 47, 51, 72, 73, 76, 86, 91, 93, 96, 115, 117, 121, 123, 131, 155, 156, 178, 204, 241, 242, 244, 270, 271]. Tematyka sieci neuronowych

była przedmiotem wielu konferencji organizowanych przez Polskie Towarzystwo Sieci Neuronowych np.: [206, 209, 219, 226, 243]. Algorytm Levenberga–Marquardta podano w pracy [68]. Gradientowe metody optymalizacji szczegółowo omawiają monografie [24, 53, 108]. Sieci neuronowe Elmana i RTRN przedstawiono w pracach [49] i [52]. Właściwości aproksymacyjne sieci neuronowych udowodnił Hornik [84, 85]. Kohonen [113] opublikował znaną monografię sieci samoorganizujących z konkurencją, natomiast zastosowania tych sieci w zadaniach kompresji obrazów przedstawiono w pracach [205, 207]. Powiązania zbiorów przybliżonych i sieci neuronowych omówiono w monografii [158]. Symulacje komputerowe dotyczące sieci typu ART i sieci Hamminga wykonano z wykorzystaniem programu NetLab dołączonego do książki [271].

Algorytmy ewolucyjne

7.1. Wprowadzenie

Inspiracją do podjęcia badań dotyczących algorytmów ewolucyjnych było naśladowanie natury. Wszystkie organizmy żywe żyją w pewnym środowisku. Posiadają specyficzny dla siebie materiał genetyczny, który zawiera informacje o nich samych i pozwala im przekazywać swoje cechy nowym pokoleniom. W trakcie reprodukcji powstaje nowy organizm, który dziedziczy pewne cechy po swoich rodzicach. Cechy te są zapisane w genach, te zaś przechowywane są w chromosomach, które z kolei składają się właśnie na materiał genetyczny — genotyp. Podczas przekazywania cech dochodzi do modyfikacji genów. Następuje krzyżowanie różniących się chromosomów obojga rodziców. Często dodatkowo zachodzi mutacja, czyli zamiana pojedynczych genów w chromosomie. Powstaje organizm różniący się od swoich rodziców, który zawiera geny swoich poprzedników, ale ma także pewne cechy charakterystyczne dla siebie. Organizm ten zaczyna żyć w danym środowisku. Jeżeli okaże się, że jest do niego dobrze przystosowany, czyli mówiąc inaczej, kombinacja genów okaże się korzystna, będzie przekazywał swój materiał genetyczny potomstwu. Osobnikowi źle przystosowanemu do środowiska trudno będzie w nim żyć i przekazywać swoje geny następnym pokoleniom.

Idea, którą przedstawiliśmy, została wykorzystana do rozwiązywania zadań optymalizacji. Okazuje się bowiem, że można zaproponować analogiczne podejście do obliczeń numerycznych, stosując tzw. *algorytmy ewolucyjne*. Środowisko definiujemy na podstawie rozwiązywanego problemu. Żyje w nim populacja osobników stanowiących potencjalne rozwiązania danego zagadnienia. Za pomocą odpowiednio zdefiniowanej funkcji przystosowania sprawdzamy, w jakim stopniu są one przystosowane do środowiska. Osobniki wymieniają między sobą materiał genetyczny, wprowadza się operatory krzyżowania i mutacji, by generować nowe rozwiązania. Spośród potencjalnych rozwiązań „przeżywają” tylko te, które są najlepiej przystosowane.

W tym rozdziale omówimy rodzinę algorytmów ewolucyjnych, tzn. klasyczny algorytm genetyczny, strategie ewolucyjne, programowanie ewolucyjne oraz programowanie genetyczne. Przedstawimy też zaawansowane techniki stosowane w algorytmach ewolucyjnych. W drugiej części rozdziału omówimy powiązania technik ewolucyjnych z sieciami neuronowymi oraz systemami rozmytymi.

7.2. Problemy optymalizacji a algorytmy ewolucyjne

W literaturze [63] wymienia się trzy typy metod poszukiwania optymalnych rozwiązań. Zaliczamy do nich metody analityczne, przeglądowe i losowe. *Metody analityczne* dzielą się na dwie klasy: metody pośrednie i metody bezpośrednie. W metodach pośrednich poszukujemy lokalnych ekstremów funkcji, rozwiązując układ równań (zwykle nieliniiowych). Układ ten otrzymujemy w wyniku przerównania gradientu funkcji celu do zera. Metody bezpośrednie poszukują lokalnego optimum przez „skakanie” po wykresie funkcji w kierunku określonym gradientem. Obie te metody nie są wolne od wad. Przede wszystkim mają one zakres lokalny, gdyż poszukują optymalnego rozwiązania w sąsiedztwie danego punktu. Ich zastosowanie uzależnione jest od istnienia pochodnych. W praktyce wiele rozwiązywanych problemów charakteryzuje się funkcjami nieciągłymi w skomplikowanej przestrzeni rozwiązań. Zatem metody analityczne mają ograniczony zakres zastosowań.

Metody przeglądowe występują w wielu postaciach. Założymy, że mamy skończoną przestrzeń poszukiwań. Najprostsza metoda polegałaby na obliczaniu wartości funkcji celu i przeglądaniu wszystkich punktów przestrzeni po kolej. Mimo swej prostoty i podobieństwa do ludzkiego rozumowania metoda ta ma jedną poważną wadę — nieefektywność. Wiele problemów cechuje się taką ogromną przestrzenią poszukiwań, że nie jest możliwe przeszukanie wszystkich punktów w rozsądny czasie.

Ostatnią z metod poszukiwań jest *metoda losowa*. Zaczęła się ona cieszyć popularnością w momencie, gdy uznano sobie słabości metod analitycznych i przeglądowych. Algorytmy poszukiwania losowego, polegające na losowym przeszukiwaniu przestrzeni i zapamiętywaniu najlepszego rozwiązania, również okazały się nieefektywne. Jednak należy je odróżnić od technik opartych na liczbach pseudolosowych i ewolucyjnym przeszukiwaniu przestrzeni rozwiązań. Algorytmy ewolucyjne są przykładem podejścia, w którym losowy wybór jest tylko narzędziem do wspomagania przeszukiwania w zakodowanej przestrzeni rozwiązań.

Algorytm ewolucyjny stanowi wzorowaną na naturalnej ewolucji metodę rozwiązywania problemów, głównie zagadnień optymalizacyjnych. Algorytmy ewolucyjne są procedurami przeszukiwania opartymi na mechanizmach doboru naturalnego i dziedziczenia. Korzystają z ewolucyjnej zasady przeżycia osobników najlepiej przystosowanych. Od tradycyjnych metod optymalizacji różnią je następujące elementy:

1. Algorytmy ewolucyjne nie przetwarzają bezpośrednio parametrów zadania, lecz ich zakodowaną postać.
2. Algorytmy ewolucyjne prowadzą przeszukiwanie, wychodząc nie z pojedynczego punktu, lecz z pewnej ich populacji.
3. Algorytmy ewolucyjne korzystają tylko z funkcji celu, nie zaś z jej pochodnych lub innych pomocniczych informacji.
4. Algorytmy ewolucyjne stosują probabilistyczne, a nie deterministyczne reguły wyboru.

Te cztery cechy, tzn. kodowanie parametrów, działanie na populacjach, korzystanie z minimum informacji o zadaniu oraz zrandomizowane operacje, składają się w efekcie

na odporność algorytmu ewolucyjnego i wynikającą stąd jego przewagę nad innymi wyżej wymienionymi technikami.

7.3. Rodzaje algorytmów zaliczanych do algorytmów ewolucyjnych

Algorytmy ewolucyjne korzystają z określeń zapożyczonych z genetyki. Mówimy na przykład o *populacji osobników*, a podstawowymi pojęciami są *gen*, *chromosom*, *genotyp*, *fenotyp*, *allel*. Używa się również odpowiadających im określeń pochodzących ze słownictwa technicznego, a więc *łańcuch*, *ciąg binarny*, *struktura*.

Populacją nazywamy zbiór *osobników* o określonej liczebności.

Osobnikami populacji w algorytmach genetycznych są zakodowane w postaci chromosomów zbiory *parametrów zadania*, czyli *rozwiązań*, określone też jako *punkty przestrzeni poszukiwań* (ang. *search points*). *Osobniki* czasami nazywa się *organizmami*.

Chromosomy — inaczej *łańcuchy* lub *ciągi kodowe* — to uporządkowane ciągi genów.

Gen — nazywany też *cechą*, *znakiem*, *detektorem* — stanowi pojedynczy element *genotypu*, w szczególności chromosomu.

Genotyp, czyli *struktura*, to zespół chromosomów danego osobnika. Zatem osobnikami populacji mogą być *genotypy* albo pojedyncze chromosomy (jeśli genotyp składa się tylko z jednego chromosomu, a tak się często przyjmuje).

Fenotyp jest zestawem wartości odpowiadających danemu genotypowi, czyli *zdekomponowaną strukturą*, a więc *zbiorem parametrów zadania* (*rozwiązań*, *punktem przestrzeni poszukiwań*).

Allel to wartość danego *genu*, określana też jako *wartość cechy* lub *wariant cechy*.

Locus to *pozycja* wskazująca miejsce położenia danego genu w *łańcuchu*, czyli chromosomie (l.mn., czyli pozycje, to *loci*).

Bardzo ważnym pojęciem w algorytmach genetycznych jest *funkcja przystosowania* (ang. *fitness function*) nazywana też *funkcją dopasowania* lub *funkcją oceny*. Stanowi ona miarę przystosowania (dopasowania) danego osobnika w populacji. Funkcja ta jest niezwykle istotna, gdyż pozwala ocenić stopień przystosowania poszczególnych osobników w populacji i na tej podstawie wybrać osobniki najlepiej przystosowane (czyli o największej wartości funkcji przystosowania), zgodnie z ewolucyjną zasadą przetrwania „najsiłniejszych” (najlepiej przystosowanych). Funkcja przystosowania również przejęła swą nazwę bezpośrednio z genetyki. Ma ona duży wpływ na działanie algorytmów ewolucyjnych i musi być odpowiednio zdefiniowana. W zagadnieniach optymalizacji funkcją przystosowania jest zwykle optymalizowana funkcja (ściślej mówiąc maksymalizowana funkcja), nazywana *funkcją celu*. W zagadnieniach minimalizacji przekształca się funkcję celu, sprowadzając problem do zagadnienia maksymalizacji. W teorii sterowania, funkcją przystosowania może być *funkcja błędu*, a w teorii gier — *funkcja kosztu*. W algorytmie ewolucyjnym, w każdej jego iteracji, ocenia się przystosowanie każdego osobnika danej populacji za pomocą funkcji przystosowania i na tej podstawie tworzy

się nową populację osobników, stanowiących zbiór potencjalnych rozwiązań problemu, np. zadania optymalizacji.

Kolejną iterację w algorytmie ewolucyjnym nazywa się *generacją*, a o nowo utworzonej populacji osobników mówi się też *nowe pokolenie* lub *pokolenie potomków*.

7.3.1. Klasyczny algorytm genetyczny

Na podstawowy (klasyczny) algorytm genetyczny, nazywany także elementarnym lub prostym algorytmem genetycznym, składają się kroki:

- 1) inicjacja, czyli wybór początkowej populacji chromosomów,
- 2) ocena przystosowania chromosomów w populacji,
- 3) sprawdzenie warunku zatrzymania,
- 4) selekcja chromosomów,
- 5) zastosowanie operatorów genetycznych,
- 6) utworzenie nowej populacji,
- 7) wyprowadzenie „najlepszego” chromosomu.

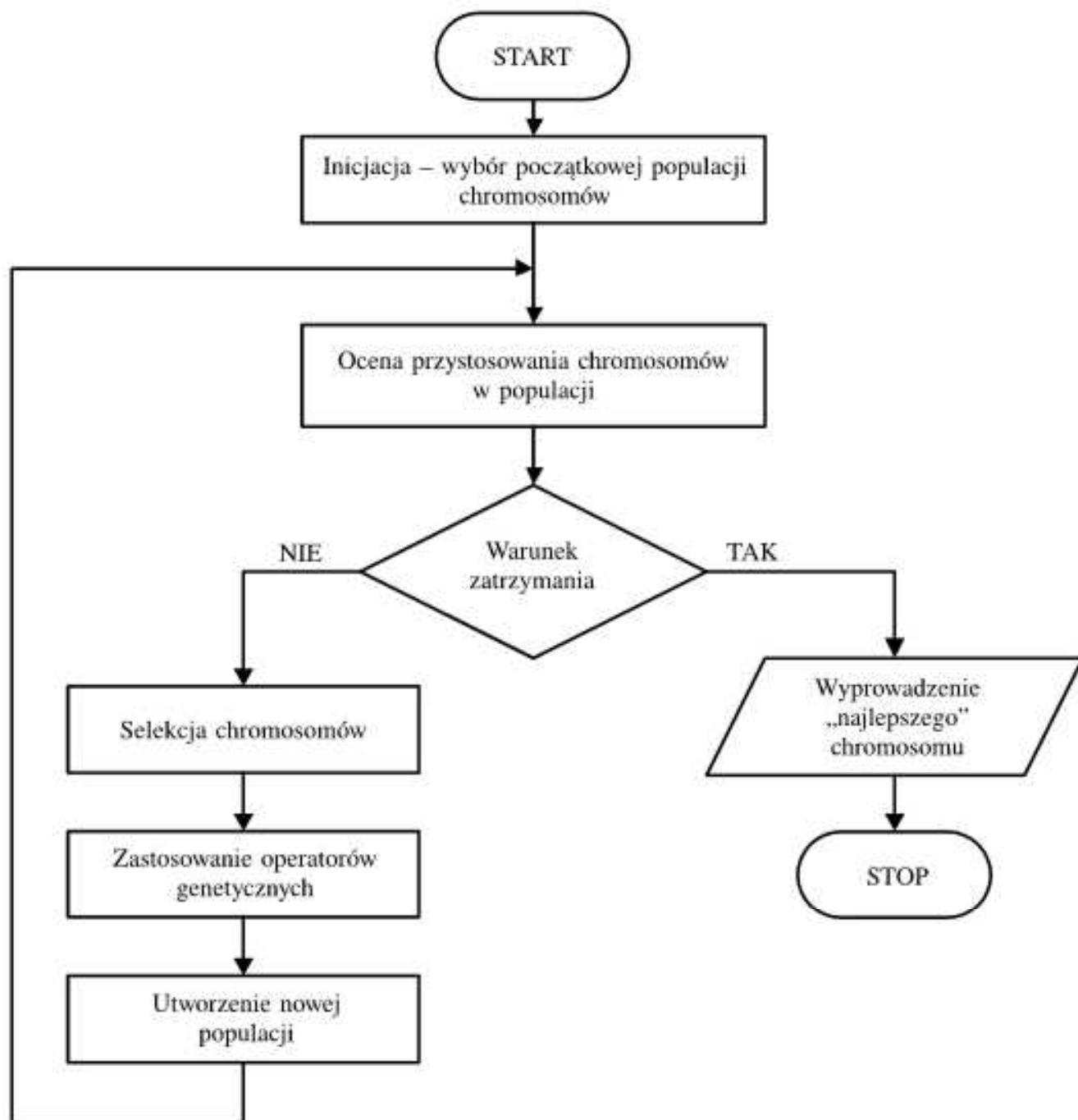
Schemat blokowy podstawowego algorytmu genetycznego przedstawia rysunek 7.1. Spróbujmy teraz dokładniej przedstawić poszczególne elementy składowe tego algorytmu.

Inicjacja, czyli utworzenie populacji początkowej, polega na losowym wyborze żądanej liczby chromosomów (osobników) reprezentowanych przez ciągi binarne o określonej długości.

Ocena przystosowania chromosomów w populacji polega na obliczeniu wartości funkcji przystosowania dla każdego chromosomu z tej populacji. Im większa jest wartość tej funkcji, tym lepsza „jakość” chromosomu. Postać funkcji przystosowania zależy od rodzaju rozwiązywanego problemu. Zakłada się, że funkcja przystosowania przyjmuje zawsze wartości nieujemne, a ponadto, że rozwiązywany problem optymalizacji jest problemem poszukiwania maksimum tej funkcji. Jeśli pierwotna postać funkcji przystosowania nie spełnia tych założeń, to dokonuje się odpowiedniej transformacji (np. problem poszukiwania minimum funkcji można łatwo sprowadzić do problemu poszukiwania maksimum).

Sprawdzenie warunku zatrzymania. Określenie warunku zatrzymania algorytmu genetycznego zależy od konkretnego zastosowania tego algorytmu. W zagadnieniach optymalizacji, jeśli znana jest wartość maksymalna (lub minimalna) funkcji przystosowania, zatrzymanie algorytmu może nastąpić po uzyskaniu żąданej wartości optymalnej, ewentualnie z określona dokładnością. Zatrzymanie algorytmu może również nastąpić, jeśli dalsze jego działanie nie poprawia już uzyskanej najlepszej wartości. Algorytm może też zostać zatrzymany po upływie określonego czasu działania lub po określonej liczbie generacji. Jeśli warunek zatrzymania jest spełniony, następuje przejście do ostatniego kroku, czyli wyprowadzenia „najlepszego” chromosomu. Jeśli nie, to następnym krokiem jest selekcja.

Selekcja chromosomów polega na wybraniu, na podstawie obliczonych wartości funkcji przystosowania (krok 2), tych chromosomów, które będą brały udział w tworze-



Rys. 7.1. Schemat blokowy algorytmu genetycznego

niu potomków do następnego pokolenia, czyli następnej generacji. Wybór ten odbywa się zgodnie z zasadą naturalnej selekcji, tzn. największe szanse na udział w tworzeniu nowych osobników mają chromosomy o największej wartości funkcji przystosowania. Istnieje wiele metod selekcji. Najbardziej popularna jest tzw. *metoda ruletki* (ang. *roulette-wheel selection*), która swą nazwę zawdzięcza analogii do losowania za pomocą koła ruletki. Każdemu chromosomowi można przypdzielić wycinek koła ruletki o wielkości proporcjonalnej do wartości funkcji przystosowania danego chromosomu. Zatem im większa jest wartość funkcji przystosowania, tym większy jest wycinek (sektor) na kole ruletki. Całe koło ruletki odpowiada sumie wartości funkcji przystosowania wszystkich chromosomów rozważanej populacji. Każdemu chromosomowi oznaczonemu przez ch_i dla $i = 1, 2, \dots, K$, gdzie K jest liczebnością populacji, odpowiada wycinek koła $v(ch_i)$ stanowiący część całego koła, wyrażony w procentach, zgodnie ze wzorem

$$v(\text{ch}_i) = p_s(\text{ch}_i) \cdot 100\%, \quad (7.1)$$

w którym

$$p_s(\text{ch}_i) = \frac{F(\text{ch}_i)}{\sum_{j=1}^K F(\text{ch}_j)}, \quad (7.2)$$

przy czym $F(\text{ch}_i)$ oznacza wartość funkcji przystosowania chromosomu ch_i , a $p_s(\text{ch}_i)$ jest prawdopodobieństwem selekcji chromosomu ch_i . Selekcja chromosomu może być widziana jako obrót kołem ruletki, w wyniku czego „wygrywa” (zostaje wybrany) chromosom należący do wylosowanego w ten sposób wycinka koła ruletki. Oczywiście im większy jest ten wycinek koła, tym większe jest prawdopodobieństwo „zwycięstwa” odpowiedniego chromosomu. Zatem prawdopodobieństwo wybrania danego chromosomu jest tym większe, im większa jest wartość jego funkcji przystosowania. Jeżeli cały okrąg koła ruletki potraktujemy jako przedział liczbowy $[0, 100]$, to wylosowanie chromosomu można potraktować jak wylosowanie liczby z zakresu $[a, b]$, gdzie a i b oznaczają, odpowiednio, początek i koniec fragmentu okręgu odpowiadającego temu wycinkowi koła; oczywiście $0 \leq a < b \leq 100$. Wówczas losowanie za pomocą koła ruletki sprowadza się do wylosowania liczby z przedziału $[0, 100]$, która odpowiada konkretnemu punktowi na okręgu koła ruletki. Inne metody zostaną przedstawione w punkcie 7.4.2.

W wyniku procesu selekcji zostaje utworzona *populacja rodzicielska*, nazywana też *pulą rodzicielską* (ang. *mating pool*), o liczebności równej K , tzn. takiej samej jak liczebność bieżącej populacji.

Zastosowanie operatorów genetycznych do chromosomów wybranych metodą selekcji prowadzi do utworzenia nowej populacji, stanowiącej populację potomków otrzymaną z populacji rodziców.

W klasycznym algorytmie genetycznym stosuje się dwa podstawowe operatory genetyczne: *operator krzyżowania* (ang. *crossover*) oraz *operator mutacji* (ang. *mutation*). Należy jednak zaznaczyć, że operator mutacji odgrywa zdecydowanie drugoplanową rolę w stosunku do operatora krzyżowania. Oznacza to, że krzyżowanie w klasycznym algorytmie genetycznym występuje prawie zawsze, podczas gdy mutacja dość rzadko. Prawdopodobieństwo wystąpienia krzyżowania przyjmuje się zwykle duże (na ogół $0,5 \leq p_k \leq 1$), natomiast zakłada się bardzo małe prawdopodobieństwo zaistnienia mutacji (często $0 \leq p_m \leq 0,1$). Wynika to także z analogii do świata organizmów żywych, gdzie mutacje zachodzą niezwykle rzadko.

W algorytmie genetycznym mutacja chromosomu może być dokonywana na populacji rodziców przed operacją krzyżowania lub na populacji potomków utworzonych w wyniku krzyżowania.

Operator krzyżowania. Pierwszym etapem krzyżowania jest wybór par chromosomów z populacji rodzicielskiej (puli rodzicielskiej). Jest to tymczasowa populacja złożona z chromosomów wybranych metodą selekcji i przeznaczonych do dalszego przetwarzania za pomocą operatorów krzyżowania i mutacji w celu utworzenia nowej populacji potomków. Na tym etapie chromosomy z populacji rodzicielskiej kojarzone są w pary. Dokonuje się tego w sposób losowy, zgodnie z prawdopodobieństwem krzyżowania p_k . Następnie dla każdej pary wybranych w ten sposób rodziców losuje się pozycję genu (locus) w chromosomie, określającą tzw. *punkt krzyżowania*. Jeżeli chromosom każdego

z rodziców składa się z L genów, to oczywiście punkt krzyżowania l_k jest liczbą naturalną mniejszą od L . Zatem wybór punktu krzyżowania sprowadza się do wylosowania liczby z przedziału $[1, L - 1]$. W wyniku krzyżowania pary chromosomów rodzicielskich otrzymuje się następującą parę potomków:

1) potomek, którego chromosom składa się z genów na pozycjach od 1 do l_k , pochodzących od pierwszego rodzica, i kolejnych genów, od pozycji $l_k + 1$ do L , pochodzących od drugiego rodzica;

2) potomek, którego chromosom składa się z genów na pozycjach od 1 do l_k , pochodzących od drugiego rodzica, i kolejnych genów, od pozycji $l_k + 1$ do L , pochodzących od pierwszego rodzica.

Przykład 7.1

Rozważmy dwa chromosomy $ch_1 = [1001001110]$ i $ch_2 = [1001111110]$, które poddajemy operacji krzyżowania. W tym przypadku chromosomy składają się z 10 genów ($L = 10$), losujemy więc liczbę całkowitą z przedziału $[1, 9]$. Założymy, że wylosowano liczbę 5. Przebieg operacji krzyżowania przedstawia się więc następująco:

$$\begin{array}{ll} \text{Para rodziców:} & \text{Para potomków:} \\ ch_1 = [10010 | 01110] & [10010 | \mathbf{11110}] \\ ch_2 = [10011 | 11110] & \xrightarrow{\text{krzyżowanie}} [10011 | \mathbf{01110}] \end{array}$$

gdzie symbolem $|$ oznaczono miejsce krzyżowania, a pogrubioną czcionką zaznaczono zamienione geny.

Operator mutacji. zgodnie z prawdopodobieństwem mutacji p_m , dokonuje zmiany wartości genu w chromosomie na przeciwną (tzn. z 0 na 1 lub z 1 na 0). Jak już wcześniej wspomniano, prawdopodobieństwo zaistnienia mutacji jest zwykle bardzo małe i oczywiście od niego zależy, czy dany gen w chromosomie podlega mutacji, czy też nie. Dokonanie mutacji zgodnie z prawdopodobieństwem p_m polega na przykład na losowaniu liczby z przedziału $[0, 1]$ dla każdego genu i wybraniu do mutacji tych genów, dla których wylosowana liczba jest równa prawdopodobieństwu p_m lub mniejsza.

Przykład 7.2

Dokonujemy operacji mutacji na chromosomie $[1001101010]$. Wartość p_m wynosi 0,02. Losujemy następujące liczby z przedziału $[0, 1]$:

0,23 0,76 0,54 0,10 0,28 0,68 0,01 0,30 0,95 0,12.

Mutacji podlega gen na pozycji 7, ponieważ wylosowana liczba losowa 0,01 jest mniejsza niż wartość prawdopodobieństwa mutacji p_m . Wobec tego, jego wartość zmienia się z 1 na 0 i otrzymujemy chromosom $[1001100010]$.

Utworzenie nowej populacji. Chromosomy otrzymane w wyniku działania operatorów genetycznych wchodzą w skład nowej populacji. Populacja ta staje się tzw. *populacją bieżącą* dla danej generacji algorytmu genetycznego. W każdej kolejnej generacji oblicza się wartości funkcji przystosowania każdego z chromosomów tej populacji. Następnie sprawdza się warunek zatrzymania algorytmu i albo wyprowadza się wynik w postaci chromosomu o największej wartości funkcji przystosowania, albo przechodzi

się do kolejnego kroku algorytmu genetycznego, tzn. selekcji. W klasycznym algorytmie genetycznym cała poprzednia populacja chromosomów zastępowana jest przez tak samo liczną nową populację potomków.

Wyprowadzenie „najlepszego” chromosomu. Jeżeli spełniony jest warunek zatrzymania algorytmu genetycznego, należy wyprowadzić wynik działania algorytmu, czyli podać rozwiązanie problemu. Najlepszym rozwiązaniem jest chromosom o największej wartości funkcji przystosowania.

Przykład 7.3

Pokażemy na prostym przykładzie, jak w praktyce działa algorytm genetyczny. Znajdziemy maksimum funkcji

$$y = 2x + 1,$$

zakładając, że x przyjmuje wartości całkowite w przedziale $[0, 31]$. W tym przypadku parametrem zadania jest x . Zbiór $\{0, 1, \dots, 31\}$ stanowi przestrzeń poszukiwań. Jest to jednocześnie zbiór potencjalnych rozwiązań zadania. Każda z 32 liczb należących do tego zbioru nazywa się punktem poszukiwań, rozwiązaniem, wartością parametru, fenotypem. Rozwiązanie optymalizujące funkcję nazywa się rozwiązaniem najlepszym lub optymalnym. Rozwiązanie zadania kodujemy binarnie (systemem dwójkowym) za pomocą pięciu bitów. Powstałe ciągi kodowe nazywane są też łańcuchami lub chromosomami. W tym przykładzie to również genotypy. Wartość genu na określonej pozycji nazywa się allelem, są to oczywiście wartości 0 lub 1. Zadanie optymalizacji polega więc na przeszukaniu przestrzeni złożonej z 32 punktów i znalezieniu tego spośród nich, dla którego funkcja przyjmuje największą wartość. Domyślamy się, że rozwiązaniem jest liczba 31, czyli chromosom zawierający same jedynek.

Zgodnie z algorytmem, rozpoczynamy od wylosowania populacji początkowej. Będziemy operować na małej populacji, liczącej osiem osobników. W wyniku losowania otrzymujemy

$$\begin{array}{ll} ch_1 = [00110], & ch_2 = [00101], \\ ch_3 = [01101], & ch_4 = [10101], \\ ch_5 = [11010], & ch_6 = [10010], \\ ch_7 = [01000], & ch_8 = [00101]. \end{array}$$

Rozpoczynamy główną pętlę algorytmu genetycznego, czyli obliczamy przystosowanie poszczególnych osobników. Dekodujemy informacje z chromosomów i otrzymujemy następujące fenotypy:

$$\begin{array}{ll} ch_1^* = 6, & ch_2^* = 5, \\ ch_3^* = 13, & ch_4^* = 21, \\ ch_5^* = 26, & ch_6^* = 18, \\ ch_7^* = 8, & ch_8^* = 5. \end{array}$$

Obliczamy przystosowanie za pomocą funkcji, która jest taka sama jak funkcja przez nas optymalizowana. Ponieważ szukamy maksimum, za najlepiej przystosowane osobniki uważa się te, które mają największą wartość funkcji przystosowania. W miejsce parametru x podstawiamy wartość fenotypu. Na przykład dla pierwszych dwóch osobników

otrzymujemy

$$F(\text{ch}_1) = 2 \cdot \text{ch}_1^* + 1 = 13, \quad F(\text{ch}_2) = 2 \cdot \text{ch}_2^* + 1 = 11.$$

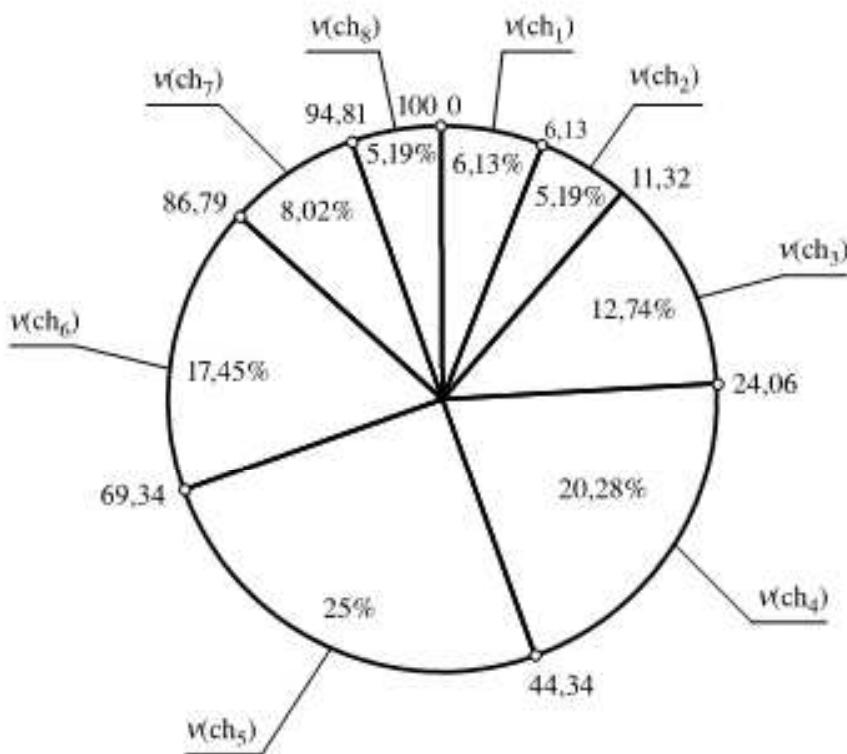
Analogicznie wyznaczamy

$$F(\text{ch}_3) = 27, \quad F(\text{ch}_4) = 43,$$

$$F(\text{ch}_5) = 53, \quad F(\text{ch}_6) = 37,$$

$$F(\text{ch}_7) = 17, \quad F(\text{ch}_8) = 11.$$

Możemy teraz w populacji wyróżnić osobniki najlepiej i najgorzej przystosowane. Jak widać, chromosom ch_5 ma największą wartość funkcji przystosowania, w przeciwnieństwie do chromosomów ch_2 i ch_8 , które mają identyczne, najmniejsze przystosowanie. Kolejnym krokiem jest selekcja chromosomów. Posłużymy się tu metodą koła ruletki.



Rys. 7.2. Koło ruletki — przykład 7.3

Na podstawie wzorów (7.1) i (7.2) otrzymujemy wycinki koła ruletki wyrażone w procentach (rys. 7.2)

$$v(\text{ch}_1) = 6,13, \quad v(\text{ch}_2) = 5,19,$$

$$v(\text{ch}_3) = 12,74, \quad v(\text{ch}_4) = 20,28,$$

$$v(\text{ch}_5) = 25, \quad v(\text{ch}_6) = 17,45,$$

$$v(\text{ch}_7) = 8,02, \quad v(\text{ch}_8) = 5,19.$$

Losowanie za pomocą koła ruletki sprowadza się do losowego wyboru liczby z przedziału $[0, 100]$ wskazującej odpowiedni wycinek na kole, a więc konkretny chromosom. Założymy, że wylosowano 8 następujących liczb:

79 44 9 74 45 86 48 23.

Oznacza to wybór następujących chromosomów:

ch₆, ch₄, ch₂, ch₆, ch₅, ch₆, ch₅, ch₄,

Jak widać, chromosom ch_5 został wylosowany dwukrotnie. Zauważmy, że jest to chromosom o największej wartości funkcji przystosowania. Ponadto trzykrotnie wylosowano chromosom ch_6 o dość dużej wartości funkcji przystosowania. Jednakże wylosowano też chromosom ch_2 o najmniejszej wartości funkcji przystosowania. Wszystkie wybrane w ten sposób chromosomy zaliczamy do tzw. *puli rodzicielskiej*.

Założymy, że żaden z chromosomów wybranych podczas selekcji nie podlega mutacji, tzn. prawdopodobieństwo $p_m = 0$. Przeprowadzimy tylko krzyżowanie, przyjmując prawdopodobieństwo krzyżowania $p_k = 0,75$. Kojarzymy osobniki w pary tak, jak są poukładane w puli rodzicielskiej. Losujemy dla każdej z nich liczbę z przedziału $[0, 1]$.

0,12 0,73 0,65 0,33,

Wszystkie wylosowane liczby są mniejsze od prawdopodobieństwa krzyżowania p_k , więc krzyżowanie zachodzi dla każdej z par. Następnie znajdujemy dla każdej pary punkty krzyżowania poprzez wylosowanie liczb całkowitych z przedziału $[1, 4]$. W efekcie otrzymujemy

Pierwsza para rodziców:

$$\begin{array}{l} \text{ch}_6 = [100\mathbf{10}] \\ \text{ch}_4 = [101\mathbf{01}] \\ l_k = 3 \end{array} \quad \xrightarrow{\text{krzyzowanie}}$$

Pierwsza para potomków:

[10001]
[10110]

Druga para rodziców:

$$\begin{array}{l} \text{ch}_2 = [00101] \quad \xrightarrow{\text{krzyżowanie}} \\ \text{ch}_6 = [10010] \\ l_c = 4 \end{array}$$

Druga para potomków:

[00100]
[10011]

Trzecia para rodziców:

$$\begin{array}{l} \text{ch}_5 = [11010] \\ \text{ch}_6 = [10010] \end{array} \quad \xrightarrow{\text{krzyzowanie}}$$

Trzecia para potomków:

[11010]
[10010]

Czwarta para rodziców:

$$\begin{array}{l} \text{ch}_5 = [11010] \\ \text{ch}_3 = [01101] \end{array} \xrightarrow{\text{krzyżowanie}} b_1 = 2$$

Czwarta para potomków:

[11101]
[01010]

W wyniku operacji krzyżowania dostajemy następującą populację potomków:

$$\begin{array}{ll} \text{Ch}_1 = [10001], & \text{Ch}_2 = [10110], \\ \text{Ch}_3 = [00100], & \text{Ch}_4 = [10011], \\ \text{Ch}_5 = [11010], & \text{Ch}_6 = [10010], \\ \text{Ch}_7 = [11101], & \text{Ch}_8 = [01010]. \end{array}$$

Chromosomy nowej populacji oznaczamy dużą literą. Przechodzimy teraz ponownie do kroku 2 algorytmu, czyli do oceny funkcji przystosowania chromosomów nowej populacji, która stanie się teraz populacją bieżącą. Dekodując informację z nowej populacji chromosomów, otrzymujemy wartości fenotypów

$$\begin{array}{ll} \text{Ch}_1^* = 17, & \text{Ch}_2^* = 22, \\ \text{Ch}_3^* = 4, & \text{Ch}_4^* = 19, \\ \text{Ch}_5^* = 26, & \text{Ch}_6^* = 18, \\ \text{Ch}_7^* = 29, & \text{Ch}_8^* = 10, \end{array}$$

a następnie wartości samej funkcji przystosowania.

$$\begin{array}{ll} F(\text{Ch}_1) = 35, & F(\text{Ch}_2) = 45, \\ F(\text{Ch}_3) = 9, & F(\text{Ch}_4) = 39, \\ F(\text{Ch}_5) = 53, & F(\text{Ch}_6) = 37, \\ F(\text{Ch}_7) = 59, & F(\text{Ch}_8) = 21. \end{array}$$

Jak widać, populacja potomków charakteryzuje się o wiele większą średnią wartością funkcji przystosowania niż populacja rodziców. Zauważmy, że w wyniku krzyżowania uzyskano chromosom Ch_7 o największej wartości funkcji przystosowania, jakiej nie miał żaden z chromosomów rodzicielskich. Mogłoby się jednak zdarzyć odwrotnie, a mianowicie po pierwszej generacji, w wyniku operacji krzyżowania, mógłby zostać „utracony” chromosom, który w populacji rodziców charakteryzował się największą wartością funkcji przystosowania. Mimo to średnie „przystosowanie” nowej populacji byłoby lepsze niż poprzedniej, a chromosomy o większej wartości funkcji przystosowania miałyby szansę pojawić się w następnych generacjach.

Przykład 7.4

Kolejny przykład ilustrujący działanie klasycznego algorytmu genetycznego będzie polegał na znalezieniu chromosomu o jak największej liczbie jedynek. Założymy, że chromosomy składają się z 9 genów, a wielkość populacji wynosi 8 chromosomów. Przyjmujemy prawdopodobieństwo krzyżowania $p_k = 0,75$ oraz prawdopodobieństwo mutacji $p_m = 0,02$. Określenie funkcji przystosowania w tym przykładzie jest bardzo proste. Odzwierciedli ona liczbę jedynek w chromosomie. Osobniki lepiej przystosowane będą miały większą liczbę jedynek, czyli wartość funkcji przystosowania będzie odpowiednio większa.

Podobnie jak w poprzednim przykładzie, rozpoczynamy od wylosowania początkowej populacji osobników

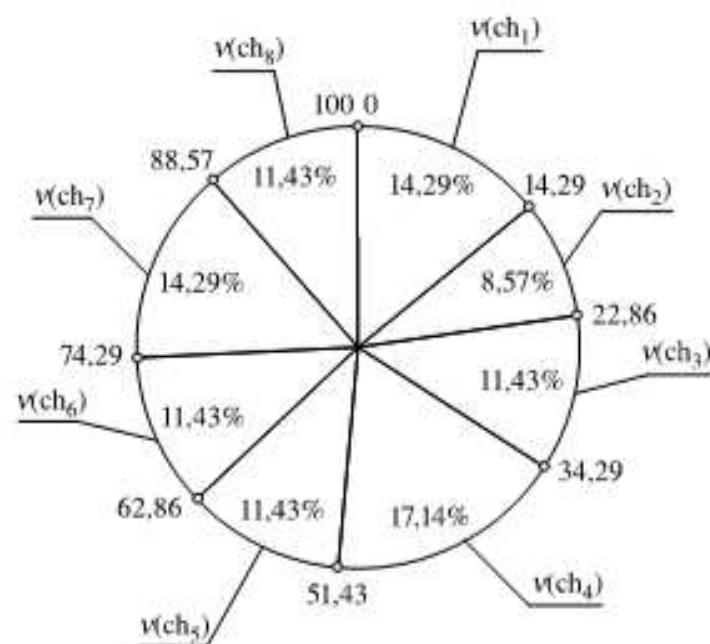
$$\begin{array}{ll} ch_1 = [100111001], & ch_2 = [001011000], \\ ch_3 = [010100110], & ch_4 = [101011110], \\ ch_5 = [110100010], & ch_6 = [100101001], \\ ch_7 = [010011011], & ch_8 = [001010011]. \end{array}$$

Przeprowadźmy symulację jednej generacji. Rozpoczynamy od wyznaczenia wartości funkcji przystosowania dla poszczególnych chromosomów.

$$\begin{array}{ll} F(ch_1) = 5, & F(ch_2) = 3, \\ F(ch_3) = 4, & F(ch_4) = 6, \\ F(ch_5) = 4, & F(ch_6) = 4, \\ F(ch_7) = 5, & F(ch_8) = 4. \end{array}$$

Jak widać, liczba jedynek i zer w chromosomach jest raczej wyrównana. Wyróżnia się tylko chromosom ch_4 , w którym jest aż 6 jedynek. Przechodzimy do realizacji następnego kroku, czyli selekcji. Zastosujemy znaną już metodę koła ruletki. Na podstawie wzorów (7.1) i (7.2) otrzymujemy następujące wycinki koła ruletki (wyrażane w procentach prawdopodobieństwa selekcji poszczególnych chromosomów):

$$\begin{array}{ll} v(ch_1) = 14,29, & v(ch_2) = 8,57, \\ v(ch_3) = 11,43, & v(ch_4) = 17,14, \\ v(ch_5) = 11,43, & v(ch_6) = 11,43, \\ v(ch_7) = 14,29, & v(ch_8) = 11,43. \end{array}$$



Rys. 7.3. Koło ruletki — przykład 7.4

Podział koła ruletki przedstawia rysunek 7.3. Dokonujemy teraz losowania 8 liczb z przedziału $[0, 100]$ w celu wybrania odpowiednich chromosomów. Założymy, że wylo-

sowano następujące liczby:

67 7 84 50 68 38 83 11.

Oznacza to wybór następujących chromosomów:

ch₆, ch₁, ch₇, ch₄, ch₆, ch₄, ch₇, ch₁.

Okazało się, że najlepiej przystosowany chromosom ch₆ został wybrany aż dwa razy, natomiast najgorszy z osobników, ch₂, nie został wybrany ani razu. Kolejnym krokiem tej generacji jest modyfikacja chromosomów za pomocą operatorów genetycznych. Rozpoczynamy od operatora krzyżowania. Założymy, że chromosomy dobrano w następujące pary:

ch₆ i ch₁, ch₇ i ch₄, ch₆ i ch₄, ch₇ i ch₁.

Losujemy liczby z przedziału [0, 1]

0,42 0,30 0,18 0,19.

a następnie porównujemy je z prawdopodobieństwem krzyżowania $p_k = 0,75$. Wszystkie wylosowane liczby są mniejsze niż prawdopodobieństwo krzyżowania, a zatem każda para podlega operacji krzyżowania. Przeprowadźmy zatem symulację tego procesu, podobnie jak w poprzednim przykładzie.

Pierwsza para rodziców:

ch₆ = [100101001]

ch₁ = [100111001]

$l_k = 4$

Pierwsza para potomków:

[100111001]

[100101001]

krzyżowanie
→

Jako miejsce krzyżowania wylosowano 4. pozycję w chromosomie. Jak widać, to krzyżowanie nie wypełniło zasadniczo na poprawę przystosowania chromosomów ch₆ i ch₁. Liczba jedynek pozostała taka sama, tzn. 4 i 5. Skrzyżujmy kolejną parę

Druga para rodziców:

ch₇ = [010011011]

ch₄ = [101011110]

$l_k = 7$

Druga para potomków:

[010011010]

[101011111]

krzyżowanie
→

Tym razem miejsce krzyżowania okazało się korzystne. Jeden z chromosomów potomnych ma już 7 jedynek. Przyjrzyjmy się kolejnym krzyżowaniom.

Trzecia para rodziców:

ch₆ = [100101001]

ch₄ = [101011110]

$l_k = 3$

Trzecia para potomków:

[100011110]

[101101001]

krzyżowanie
→

Czwarta para rodziców:

ch₇ = [010011011]

ch₁ = [100111001]

$l_k = 5$

Czwarta para potomków:

[010011001]

[100111011]

krzyżowanie
→

Zobaczmy, jak po operacji krzyżowania wygląda nowa populacja. Osobniki nowej populacji oznaczone są dużą literą.

$$\begin{array}{ll} \text{Ch}_1 = [100111001], & \text{Ch}_2 = [100101001], \\ \text{Ch}_3 = [010011010], & \text{Ch}_4 = [101011111], \\ \text{Ch}_5 = [100011110], & \text{Ch}_6 = [101101001], \\ \text{Ch}_7 = [010011001], & \text{Ch}_8 = [100111011]. \end{array}$$

Mutację przeprowadzamy już na nowej populacji. Jak pamiętamy, prawdopodobieństwo mutacji wynosi $p_m = 0,02$. Dla każdego z genów losujemy liczbę z zakresu $[0, 1]$ i sprawdzamy, czy jest mniejsza od prawdopodobieństwa p_m . Mutacja polega na zamianie wartości genu na wartość przeciwną. Zobaczmy, jak przebiegły ten proces

$$\begin{array}{ll} \text{Ch}_1 = [100111001], & \text{Ch}_2 = [100101001], \\ \text{Ch}_3 = [010011010], & \text{Ch}_4 = [111011111], \\ \text{Ch}_5 = [100011110], & \text{Ch}_6 = [101101101], \\ \text{Ch}_7 = [010011001], & \text{Ch}_8 = [100111011]. \end{array}$$

Prawdopodobieństwo mutacji jest tak małe, że w całej populacji wystąpiło tylko w dwóch miejscach, tzn. w osobnikach Ch_4 i Ch_6 . Sprawdźmy teraz wartość funkcji przystosowania poszczególnych osobników po jednej generacji.

$$\begin{array}{ll} F(\text{Ch}_1) = 5, & F(\text{Ch}_2) = 4, \\ F(\text{Ch}_3) = 4, & F(\text{Ch}_4) = 8, \\ F(\text{Ch}_5) = 5, & F(\text{Ch}_6) = 6, \\ F(\text{Ch}_7) = 4, & F(\text{Ch}_8) = 6. \end{array}$$

Jak widać, po jednej generacji udało nam się znaleźć osobnika o większej wartości funkcji przystosowania. Poza tym poprawiła się też średnia wartość funkcji przystosowania całej populacji. Osobniki lepiej przystosowane będą miały szanse częściej pojawiać się w kolejnych generacjach i dać początek nowym, lepiej przystosowanym.

Przykład 7.5

Zastosujmy teraz klasyczny algorytm genetyczny w tzw. „problemie plecakowym”. Wyobraźmy sobie plecak, który ma pewną określoną nośność W . Można do niego włożyć n przedmiotów, o wadze w_i oraz ważności p_i , $i = 1, \dots, n$. Zadanie polega na takim załadunku plecaka, aby znalazły się w nim przedmioty, z których mielibyśmy jak największą korzyść (czyli uzyskali maksymalną możliwą sumę ważności p_i umieszczonych przedmiotów). Oczywiście, nie możemy przekroczyć nośności W , tzn.

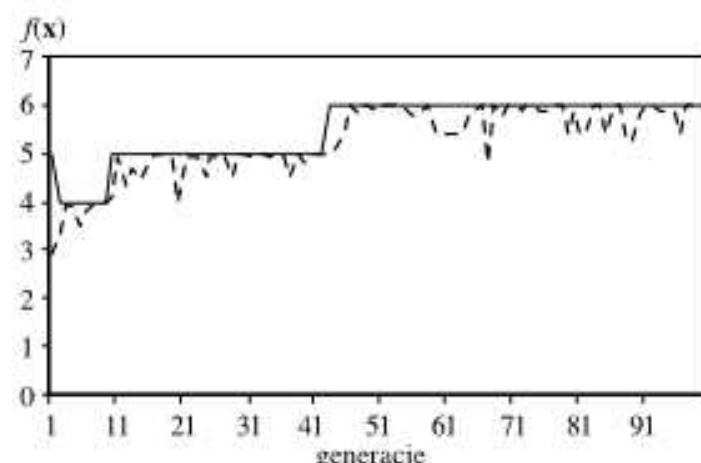
$$\sum_{i=1}^n w_i x_i \leq W. \quad (7.3)$$

Bez uwzględnienia warunku (7.3) najlepsze okazywałyby się chromosomy odpowiadające występowaniu wszystkich przedmiotów w plecaku. Rozwiązaniem tego problemu jest nałożenie kary na takie chromosomy. W tym przykładzie metodą karania osobników nie spełniających określonych wymagań będzie przyjęcie zerowej wartości funkcji przystosowania. Takie osobniki nie mają szans na reprodukcję.

Spróbujmy teraz zakodować rozwiążanie, wykorzystując zapis chromosomu binarnego. Wartości 0 i 1 kolejnego genu x_i odpowiadająby brakowi lub występowaniu i -tej rzeczy w plecaku. Funkcja przystosowania przybiera postać

$$f(\mathbf{x}) = \sum_{i=1}^n p_i x_i. \quad (7.4)$$

W naszym zadaniu bierzymy pod uwagę $n = 10$ przedmiotów, każdy o ważności $p_i = 1$ oraz wagę kolejno od 1 do 10, czyli $w_i = i$, $i = 1, \dots, 10$. Ponieważ ważność wszystkich przedmiotów jest taka sama, rozwiążanie sprowadza się do umieszczenia jak największej ich liczby w plecaku. Zakładamy, że nośność wynosi 27, czyli jest równa prawie połowie sumy wag wszystkich przedmiotów. Do poszukiwania optymalnego zapelnienia plecaka zastosujemy klasyczny algorytm genetyczny z selekcją proporcjonalną (koło ruletki) oraz prawdopodobieństwem krzyżowania $p_k = 0,7$, prawdopodobieństwem mutacji $p_m = 0,01$. Liczba osobników w populacji wynosi 10. Prześledźmy zmiany wartości funkcji przystosowania osobników dla 100 kolejnych generacji algorytmu (rys. 7.4).



Rys. 7.4. Wykres maksymalnej (linia ciągła) oraz średniej (linia przerywana) w populacji wartości funkcji przystosowania w 100 kolejnych generacjach

Wykres odzwierciedla w kolejnych generacjach zmianę średniej wartości przystosowania osobników (linia przerywana). Wartości te nie odbiegały znacznie od wyników najlepszego chromosomu (linia ciągła). Można zatem wywnioskować, że różnorodność populacji nie była zbyt duża. W 43. generacji znalezione zostało najlepsze rozwiązanie i poprawiło się przystosowanie całej populacji. Rozwiążaniem problemu okazało się sześć przedmiotów o wagie 1, 2, 4, 5, 7 i 8 umieszczonych w plecaku.

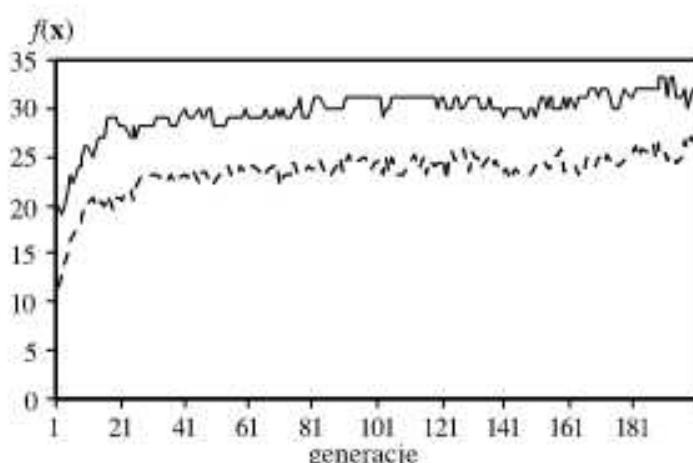
Powyższy algorytm został uruchomiony 100 razy. Otrzymano:

- 2 rozwiązania z 4 przedmiotami,
- 24 rozwiązania z 5 przedmiotami,
- 74 rozwiązania z 6 przedmiotami.

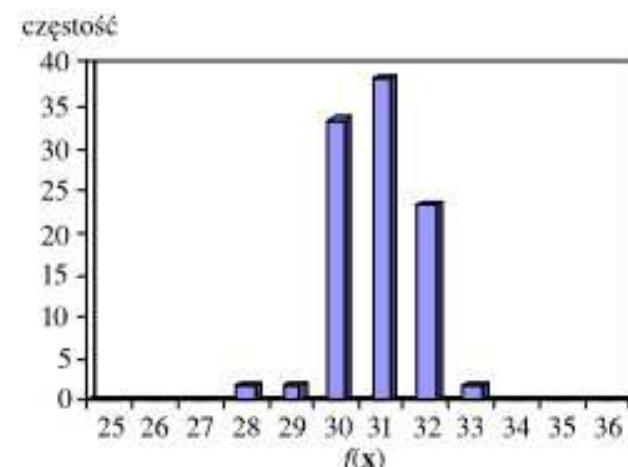
Przykład 7.6

Ponownie rozwiążemy problem plecakowy przedstawiony w przykładzie 7.5. Tym razem założymy, że $n = 50$, natomiast nośność plecaka W wynosi 318, co jest prawie połową sumy wag wszystkich przedmiotów. Przyjmujemy $p_i = 1$ oraz $w_i = i$, identycznie jak w przykładzie 7.5. Wielkość populacji, ze względu na skomplikowanie zadania, ustalmy

na 100 osobników. Stosujemy ten sam algorytm genetyczny, ale liczba generacji wynosi 200. Rysunek 7.5 ilustruje działanie algorytmu, natomiast rysunek 7.6 przedstawiaczęstość występowania różnych rozwiązań w 100 uruchomieniach algorytmu genetycznego. Z analizy histogramu (rys. 7.6) wynika, że znaleziono 38 rozwiązań gwarantujących upakowanie 31 przedmiotów, a tylko 2 rozwiązania z 33 przedmiotami w plecaku. Widać, że problem ten jest bardziej złożony aniżeli opisany w przykładzie 7.5. Chromosom ma teraz znacznie większą długość i warto zastanowić się nad sposobem poprawy działania algorytmu, co zrobimy w przykładzie 7.16.



Rys. 7.5. Wykres maksymalnej (linia ciągła) oraz średniej (linia przerywana) w populacji wartości funkcji przystosowania w 200 kolejnych generacjach



Rys. 7.6. Histogram rozwiązań uzyskiwanych w 100 przeprowadzonych próbach algorytmu genetycznego

7.3.1.1. Teoretyczne podstawy działania algorytmów genetycznych

Aby lepiej zrozumieć działanie algorytmu genetycznego, przeprowadzimy rozważania, opierając się na pojęciu *schematu*, i przedstawimy podstawowe twierdzenie dotyczące algorytmów genetycznych, tzw. *twierdzenie o schematach*. Pojęcie schematu zostało wprowadzone w celu określenia zbioru chromosomów o pewnych wspólnych cechach, podobieństwach. Jeżeli allele przyjmują wartości 0 lub 1, czyli rozważamy chromosomy o binarnym alfabetie, to schemat jest zbiorem chromosomów zawierających zera i jedynki na wyszczególnionych pozycjach. Schematy wygodnie jest rozpatrywać, korzystając z rozszerzonego alfabetu $\{0, 1, *\}$, w którym oprócz 0 i 1 wprowadzono dodatkowo symbol * w celu określenia dowolnej spośród tych wartości, tzn. albo 0, albo 1; symbol * na danej pozycji oznacza „wszystko jedno” (ang. *don't care*).

Przykład 7.7

Poniżej podajemy 2 przykłady schematów

$$10*1 = \{1001, 1011\},$$

$$*01*10 = \{001010, 001110, 101010, 101110\}.$$

Mówimy, że *chromosom należy do danego schematu*, jeżeli dla każdej pozycji (locus) $j = 1, 2, \dots, L$, gdzie L jest długością chromosomu, symbol występujący na j -ej pozycji chromosomu odpowiada symbolowi na j -ej pozycji schematu, przy czym zarówno 0,

jak 1 odpowiada symbolowi *. To samo można wyrazić, mówiąc, że *chromosom pasuje do schematu* (inaczej jest *dopasowany do schematu*) lub jest *reprezentantem schematu*. Zauważmy, że jeżeli w schemacie występuje m symboli *, to schemat ten zawiera 2^m chromosomów. Ponadto każdy chromosom (łańcuch) o długości L należy do 2^L schematów.

Przykład 7.8

Łańcuch 01 pasuje do 4 schematów: **, *1, 0*, 01, a łańcuch 100 pasuje do 8 schematów: ***, **0, *0*, 1**, *00, 1*0, 10*, 100.

Algorytm genetyczny opiera się na zasadzie przetwarzania osobników (chromosomów) najlepiej przystosowanych. Niech $\mathbf{P}(0)$ oznacza początkową populację osobników, a $\mathbf{P}(t)$ populację bieżącą w generacji t działania algorytmu. Z każdej populacji $\mathbf{P}(t)$, $t = 0, 1, \dots$ wybiera się metodą selekcji chromosomy o najlepszym przystosowaniu do puli rodzicielskiej (ang. *mating pool*) oznaczonej przez $\mathbf{M}(t)$. Następnie, kojarząc w pary osobniki rodzicielskie z populacji $\mathbf{M}(t)$ i dokonując operacji krzyżowania zgodnie z prawdopodobieństwem krzyżowania p_k oraz operacji mutacji zgodnie z prawdopodobieństwem mutacji p_m , otrzymujemy nową populację $\mathbf{P}(t+1)$, do której wchodzą potomkowie osobników z populacji $\mathbf{M}(t)$.

Chcielibyśmy, aby liczba chromosomów pasujących do schematu reprezentującego dobre rozwiązań w populacji $\mathbf{P}(t)$ wzrastała wraz ze wzrostem liczby generacji t .

Na odpowiednie przetwarzanie schematów w algorytmie genetycznym mają wpływ 3 czynniki: selekcja chromosomów, krzyżowanie i mutacja. Przeanalizujemy zatem działanie każdego z nich, badając ich wpływ na oczekiwany liczbę reprezentantów określonego schematu.

Niech S będzie danym schematem, a $c(S, t)$ liczbą chromosomów w populacji $\mathbf{P}(t)$ pasujących do schematu S . Wobec tego $c(S, t)$ jest liczbą elementów zbioru $\mathbf{P}(t) \cap S$.

Zacznijmy od zbadania wpływu selekcji. Podczas selekcji chromosomy z populacji $\mathbf{P}(t)$ kopowane są do puli rodzicielskiej $\mathbf{M}(t)$ z prawdopodobieństwem danym wzorem (7.2). Niech $F(S, t)$ oznacza średnią wartość funkcji przystosowania chromosomów w populacji $\mathbf{P}(t)$, pasujących do schematu S . Jeżeli

$$\mathbf{P}(t) \cap S = \{\text{ch}_i, \dots, \text{ch}_{c(S, t)}\},$$

to

$$F(S, t) = \frac{\sum_{i=1}^{c(S, t)} F(\text{ch}_i)}{c(S, t)}. \quad (7.5)$$

$F(S, t)$ nazywa się także *przystosowaniem schematu S w generacji t*.

Niech $\mathfrak{F}(t)$ oznacza sumę wartości funkcji przystosowania chromosomów w populacji $\mathbf{P}(t)$ o liczbowości K , tzn.

$$\mathfrak{F}(t) = \sum_{i=1}^K F(\text{ch}_i^{(t)}). \quad (7.6)$$

Oznaczmy przez $\overline{F}(t)$ średnią wartość funkcji przystosowania chromosomów w tej populacji, czyli

$$\overline{F}(t) = \frac{1}{K} \mathfrak{F}(t). \quad (7.7)$$

Niech $\text{chr}^{(t)}$ będzie elementem puli rodzicielskiej $\mathbf{M}(t)$. Dla każdego $\text{chr}^{(t)} \in \mathbf{M}(t)$ i dla każdego $i = 1, \dots, c(S, t)$ prawdopodobieństwo, że $\text{chr}^{(t)} = \text{ch}_i$ jest dane wzorem $F(\text{ch}_i)/\mathfrak{I}(t)$. Zatem oczekiwana liczba chromosomów w populacji $\mathbf{M}(t)$ równych ch_i wynosi

$$K \frac{F(\text{ch}_i)}{\mathfrak{I}(t)} = \frac{F(\text{ch}_i)}{\bar{F}(t)}.$$

Oczekiwana liczba chromosomów w zbiorze $\mathbf{P}(t) \cap S$ wybranych do puli rodzicielskiej $\mathbf{M}(t)$ jest zatem równa

$$\sum_{i=1}^{c(S, t)} \frac{F(\text{ch}_i)}{\bar{F}(t)} = c(S, t) \frac{F(S, t)}{\bar{F}(t)},$$

co wynika ze wzoru (7.5). Skoro każdy chromosom z puli rodzicielskiej $\mathbf{M}(t)$ jest jednocześnie chromosomem należącym do populacji $\mathbf{P}(t)$, to chromosomy ze zbioru $\mathbf{M}(t) \cap S$ są po prostu tymi samymi chromosomami, które ze zbioru $\mathbf{P}(t) \cap S$ zostały wybrane do populacji $\mathbf{M}(t)$. Jeżeli przez $b(S, t)$ oznaczymy liczbę chromosomów z puli rodzicielskiej $\mathbf{M}(t)$ pasujących do schematu S , czyli liczbę elementów zbioru $\mathbf{M}(t) \cap S$, to z powyższych rozważań otrzymujemy następujący wniosek.

Wniosek 7.1 (wpływ selekcji)

Wartość oczekiwana $b(S, t)$, czyli oczekiwana liczba chromosomów w puli rodzicielskiej $\mathbf{M}(t)$ pasujących do schematu S , jest określona wzorem

$$E[b(S, t)] = c(S, t) \frac{F(S, t)}{\bar{F}(t)}. \quad (7.8)$$

Wynika stąd, że jeżeli schemat S zawiera chromosomy o wartości funkcji przystosowania powyżej średniej, czyli przystosowanie schematu S w generacji t jest większe niż średnia wartość funkcji przystosowania chromosomów w populacji $\mathbf{P}(t)$, co oznacza, że $F(S, t)/\bar{F}(t) > 1$, to oczekiwana liczba chromosomów pasujących do schematu S w puli rodzicielskiej $\mathbf{M}(t)$ jest większa niż liczba chromosomów pasujących do schematu S populacji $\mathbf{P}(t)$. Można zatem stwierdzić, że selekcja powoduje rozprzestrzenianie się schematów o przystosowaniu „lepszym” od przeciętnego, a zanikanie schematów o „gorszym” przystosowaniu.

Zanim przystąpimy do analizy wpływu działania operatorów genetycznych, tzn. krzyżowania i mutacji, na chromosomy z puli rodzicielskiej, zdefiniujemy potrzebne do dalszych rozważań pojęcie *rzędu* oraz *rozpiętości* schematu. Niech L oznacza długość chromosomów należących do schematu S .

Definicja 7.1

Rzqd (ang. *order*) schematu S , nazywany także *licznością* schematu i oznaczany przez $o(S)$, jest to liczba ustalonych pozycji w schemacie, tzn. zer i jedynek w przypadku alfabetu $\{0, 1, *\}$.

Przykład 7.9

Poniżej wyznaczamy rzędy 4 schematów:

$$o(10*1) = 3, \quad o(*01*10) = 4, \quad o(**0*1*) = 2, \quad o(*101**) = 3.$$

Rząd $o(S)$ schematu równa się długości L minus liczba symboli $*$, co łatwo sprawdzić na powyższych przykładach (dla $L = 4$ z jednym symbolem $*$ oraz dla $L = 6$ z dwoma, czterema i trzema symbolami $*$). Łatwo zauważyc, że rząd schematu o samych symbolach $*$ jest równy zero, tzn. $o(****) = 0$, a rząd schematu bez żadnego symbolu $*$ jest równy L , np. $o(10011010) = 8$. Rząd schematu $o(S)$ jest liczbą całkowitą z przedziału $[0, L]$.

DEFINICJA 7.2

Rozpiętość (ang. *defining length*) schematu S , nazywana także *długością* schematu (nie należy mylić z długością L), oznaczana przez $d(S)$, jest to odległość między pierwszym i ostatnim ustalonym symbolem (tj. różnica między prawą i lewą skrajną pozycją o ustalonym symbolu).

Przykład 7.10

Rozpiętości schematów podanych w przykładzie 7.9 są następujące:

$$\begin{aligned} d(10*1) &= 4 - 1 = 3, & d(*01*10) &= 6 - 2 = 4, \\ d(**0*1*) &= 5 - 3 = 2, & d(*101**) &= 4 - 2 = 2. \end{aligned}$$

Rozpiętość schematu $d(S)$ jest liczbą całkowitą z przedziału $[0, L - 1]$. Zauważmy, że rozpiętość schematu z ustalonimi symbolami na pierwszej i ostatniej pozycji równa się $L - 1$ (jak w pierwszym z powyższych przykładów). Rozpiętość schematu z jedną ustaloną pozycją równa się zeru, np. $d(**1*) = 0$. Rozpiętość schematu określa zwartość informacji zawartej w schemacie.

Przejdźmy teraz do rozważań na temat wpływu operacji krzyżowania na przetwarzanie schematów w algorytmie genetycznym. Zauważmy najpierw, że niektóre schematy są bardziej wrażliwe na zniszczenie podczas krzyżowania niż inne. Przyjrzyjmy się na przykład schematom $S_1 = 1****0*$ i $S_2 = **01***$ oraz chromosomowi $ch = [1001101]$ pasującemu do obu schematów. Widać, że schemat S_2 ma większą szansę przetrwania operacji krzyżowania niż schemat S_1 , który jest bardziej narażony na „rozcięcie” w punkcie krzyżowania 1, 2, 3, 4 lub 5. Schemat S_2 można rozdzielić, tylko wybierając punkt krzyżowania równy 3. Zwróćmy uwagę na rozpiętość obu schematów, która — jak widać — jest istotna w procesie krzyżowania.

Analizując wpływ operacji krzyżowania na pulę rodzicielską $\mathbf{M}(t)$, rozważmy dany chromosom ze zbioru $\mathbf{M}(t) \cap S$, czyli chromosom z puli rodzicielskiej pasujący do schematu S . Prawdopodobieństwo, że chromosom ten zostanie wybrany do krzyżowania, wynosi p_k . Jeżeli żaden z potomków tego chromosomu nie będzie należał do schematu S , to oznacza, że punkt krzyżowania musi znajdować się między pierwszym i ostatnim ustalonym symbolem w schemacie S . Prawdopodobieństwo tego równa się $d(S)/(L-1)$. Wynikają stąd kolejne wnioski.

Wniosek 7.2 (wpływ krzyżowania)

Dla danego chromosomu w $\mathbf{M}(t) \cap S$ prawdopodobieństwo, że chromosom ten zostanie wybrany do krzyżowania i żaden z jego potomków nie będzie należał do schematu S , jest ograniczone z góry przez

$$p_k \frac{d(S)}{L-1}.$$

Wielkość tę nazywamy *prawdopodobieństwem zniszczenia schematu S* .

Wniosek 7.3

Dla danego chromosomu w $\mathbf{M}(t) \cap S$ prawdopodobieństwo, że chromosom ten albo nie zostanie wybrany do krzyżowania, albo co najmniej jeden z jego potomków będzie należał do schematu S po krzyżowaniu, jest ograniczone z dołu przez

$$1 - p_k \frac{d(S)}{L-1}.$$

Wielkość tę nazywamy *prawdopodobieństwem przetrwania schematu S* .

Łatwo pokazać, że gdy dany chromosom należy do schematu S i jest on wybrany do krzyżowania oraz drugi chromosom rodzicielski też należy do schematu S , wtedy obydwa chromosomy będące ich potomkami także należą do schematu S . Wniosek 7.2 i 7.3 potwierdza znaczącą rolę rozpiętości schematu $d(S)$ w prawdopodobieństwie zniszczenia lub przetrwania schematu.

Rozważmy teraz wpływ mutacji na pulę rodzicielską $\mathbf{M}(t)$. Operator mutacji zmienia losowo z prawdopodobieństwem p_m wartość na określonej pozycji z 0 na 1 lub odwrotnie. Jest oczywiste, że jeżeli schemat ma przetrwać mutację, to wszystkie ustalone pozycje w schemacie muszą pozostać niezmienione. Dany chromosom z puli rodzicielskiej należący do schematu S , czyli chromosom ze zbioru $\mathbf{M}(t) \cap S$, pozostaje w schemacie S wtedy i tylko wtedy, gdy żaden z symboli w tym chromosomie, odpowiadających ustalonemu symbolowi w schemacie S , nie ulega zmianie podczas mutacji. Prawdopodobieństwo takiego zdarzenia wynosi $(1 - p_m)^{o(S)}$. Rezultat ten przedstawimy w postaci kolejnego wniosku.

Wniosek 7.4 (wpływ mutacji)

Dla danego chromosomu w $\mathbf{M}(t) \cap S$ prawdopodobieństwo, że chromosom ten będzie należał do schematu S po operacji mutacji, jest dane przez

$$(1 - p_m)^{o(S)}. \quad (7.9)$$

Wielkość tę nazywamy *prawdopodobieństwem przetrwania mutacji* przez schemat S .

Wniosek 7.5

Jeżeli prawdopodobieństwo mutacji p_m jest małe ($p_m \ll 1$), to można przyjąć, że prawdopodobieństwo przetrwania mutacji przez schemat S , określone we wniosku 7.4, jest w przybliżeniu równe

$$1 - p_m o(S). \quad (7.10)$$

Efekt połączenia wpływu selekcji, krzyżowania i mutacji (wnioski 7.1–7.4), przy uwzględnieniu faktu, że jeśli chromosom ze zbioru $\mathbf{M}(t) \cap S$ daje potomka pasującego do schematu S , to należy on do $\mathbf{P}(t+1) \cap S$, prowadzi do otrzymania następującego schematu reprodukcji [24]:

$$E[c(S, t+1)] \geq c(S, t) \frac{F(S, t)}{\bar{F}(t)} \left(1 - p_k \frac{d(S)}{L-1}\right) (1 - p_m)^{o(S)}. \quad (7.11)$$

Zależność (7.11) pokazuje, jak liczba chromosomów pasujących do danego schematu zmienia się z populacji na populację. Trzy czynniki, odzwierciedlone po prawej stronie wyrażenia (7.11), mają wpływ na tę zmianę, a mianowicie: $F(S, t)/\bar{F}(t)$ wskazujący

rolę średniej wartości funkcji przystosowania, $1 - p_k d(S)/(L - 1)$ wskazującą wpływ krzyżowania oraz $(1 - p_m)^{o(S)}$ wskazującą wpływ mutacji. Im większa jest wartość każdego z tych czynników, tym większa jest oczekiwana liczba dopasowań do schematu S w następnej populacji. Wniosek 7.5 pozwala przedstawić zależność (7.11) w postaci

$$E[c(S, t + 1)] \geq c(S, t) \frac{F(S, t)}{\bar{F}(t)} \left(1 - p_k \frac{d(S)}{L - 1} - p_m o(S) \right). \quad (7.12)$$

Dla dużych populacji wzór (7.12) można aproksymować przez

$$c(S, t + 1) \geq c(S, t) \frac{F(S, t)}{\bar{F}(t)} \left(1 - p_k \frac{d(S)}{L - 1} - p_m o(S) \right). \quad (7.13)$$

Z zależności (7.11) i (7.12) wynika, że oczekiwana liczba chromosomów pasujących do schematu S w następnej generacji jest funkcją aktualnej liczby chromosomów należących do tego schematu, względnego przystosowania schematu oraz rzędu i rozpiętości schematu. Widać, że schematy o przystosowaniu powyżej średniej oraz małego rzędu i o małej rozpiętości charakteryzują się rosnącą liczbą swoich reprezentantów w kolejnych populacjach. Wzrost ten jest wykładniczy, co wynika z zależności (7.8), którą dla dużych populacji można zastąpić przybliżoną zależnością rekurencyjną postaci [136]

$$c(S, t + 1) = c(S, t) \frac{F(S, t)}{\bar{F}(t)}. \quad (7.14)$$

Jeżeli założymy, że schemat S ma przystosowanie o $\varepsilon\%$ powyżej średniej, tzn.

$$F(S, t) = \bar{F}(t) + \varepsilon \bar{F}(t), \quad (7.15)$$

to podstawiając zależność (7.13) do nierówności (7.12) oraz zakładając, że ε nie zmienia się w czasie, i startując od $t = 0$, otrzymujemy

$$c(S, t) = c(S, 0)(1 + \varepsilon)^t$$

i

$$\varepsilon = (F(S, t) - \bar{F}(t)) / \bar{F}(t), \quad (7.16)$$

tzn. $\varepsilon > 0$ dla schematu o przystosowaniu powyżej średniej i $\varepsilon < 0$ dla schematu o przystosowaniu poniżej średniej.

Równanie (7.16) opisuje postęp geometryczny. Wynika z niego, że w procesie reprodukcji schematy lepsze (gorsze) od przeciętnej są wybierane w liczbie rosnącej (malejącej) wykładniczo w kolejnych generacjach algorytmu genetycznego. Zauważmy, że zależności (7.8)–(7.14) oparte są na założeniu, że funkcja przystosowania F przyjmuje tylko wartości dodatnie. Gdy stosuje się algorytmy genetyczne do problemów optymalizacyjnych, w których optymalizowana funkcja może przyjmować wartości ujemne, wymagane są pewne dodatkowe odwzorowania między optymalizowaną funkcją a funkcją przystosowania. Końcowy rezultat wynikający z zależności (7.11)–(7.13) można sformułować w formie twierdzenia. Jest to podstawowe twierdzenie algorytmów genetycznych, czyli *twierdzenie o schematach* [82].

TWIERDZENIE 7.1

Schematy małego rzędu, o małej rozpiętości i o przystosowaniu powyżej średniej otrzymują rosnącą wykładniczo liczbę swoich reprezentantów w kolejnych generacjach algorytmu genetycznego.

Wobec powyższego twierdzenia ważnym zagadnieniem jest kodowanie, które powinno dawać dobre schematy małego rzędu, o małej rozpiętości i o przystosowaniu powyżej średniej. Jako rezultat pośredni twierdzenia 7.1 można przyjąć następującą hipotezę. Jest to tzw. *hipoteza cegiełek* (lub *bloków budujących*) [63, 136].

HIPOTEZA 7.1

Algorytm genetyczny dąży do osiągnięcia rezultatu bliskiego optimum poprzez zestawienie dobrych schematów (o przystosowaniu powyżej średniej), małego rzędu i o małej rozpiętości. Schematy te są nazywane *cegielkami* (lub blokami budującymi).

Hipotezę bloków budujących wysunięto na podstawie twierdzenia o schematach, wnioskując, że algorytmy genetyczne badają przestrzeń poszukiwań za pomocą schematów małego rzędu i o małej rozpiętości, które następnie biorą udział w wymianie informacji podczas krzyżowania.

Chociaż pewne badania zostały poczynione w kierunku udowodnienia tej hipotezy, to jednak w przypadku większości nietrywialnych zastosowań opieramy się głównie na rezultatach empirycznych. W ciągu ostatnich kilkunastu lat powstało wiele prac dotyczących zastosowań algorytmów genetycznych popierających tę hipotezę. Przy założeniu słuszności hipotezy problem kodowania wydaje się krytyczny dla działania algorytmu genetycznego; kodowanie powinno spełniać koncepcję małych bloków budujących. Siła, jaką reprezentują algorytmy genetyczne, zyskując korzystną przewagę nad innymi tradycyjnymi metodami, tkwi niewątpliwie w przetwarzaniu wielkiej liczby schematów.

7.3.2. Strategie ewolucyjne

Poznanie idei strategii ewolucyjnych najlepiej rozpocząć od porównania ich z działaniem algorytmów genetycznych. Głównym podobieństwem jest oczywiście to, że obie metody operują na populacjach potencjalnych rozwiązań oraz korzystają z zasady selekcji i przetwarzania osobników najlepiej przystosowanych. Jest jednak wiele różnic między tymi dwoma algorytmami. Pierwsza różnica dotyczy sposobu reprezentacji osobników. Strategie ewolucyjne działają na wektorach liczb zmiennoprzecinkowych, podczas gdy klasyczne algorytmy genetyczne działają na wektorach binarnych.

Druga różnica między strategiami ewolucyjnymi a algorytmami genetycznymi ukryta jest w procesie selekcji. W algorytmie genetycznym do nowej populacji wybierana jest pewna liczba osobników, odpowiadająca wielkości populacji rodzicielskiej. Następuje to poprzez losowanie, podczas którego szansa wyboru osobnika uzależniona jest od wartości jego funkcji przystosowania. Powoduje to, że najgorsze chromosomy również mogą zostać wylosowane. Natomiast w strategiach ewolucyjnych, w procedurze selekcji tworzona jest tymczasowa populacja, a jej wielkość różni się od wielkości populacji rodzicielskiej. Kolejna generacja osobników powstaje przez wybór najlepszych. Niezależnie

od zastosowanej w algorytmie genetycznym metody selekcji (metoda „koła ruletki” czy też np. metoda rankingowa) lepiej przystosowane osobniki mogą być wybierane kilkakrotnie. W strategiach ewolucyjnych osobniki wybierane są bez powtórzeń. Selekcja jest deterministyczna.

Trzecia różnica między strategiami ewolucyjnymi a algorytmami genetycznymi dotyczy kolejności procedur selekcji i rekombinacji (czyli zmiany genów poprzez działanie operatorów genetycznych). W strategiach ewolucyjnych najpierw następuje proces rekombinacji, a następnie selekcji. Potomek jest rezultatem krzyżowania dwóch osobników rodzicielskich oraz mutacji, przy czym w niektórych wersjach strategii ewolucyjnych stosuje się jedynie mutację. Utworzona w ten sposób populacja tymczasowa, o której już wspomniano, podlega procedurze selekcji, która redukuje rozmiar tej populacji do rozmiaru populacji rodziców. W algorytmach genetycznych jest odwrotnie. Najpierw odbywa się selekcja osobników, na których później dopiero stosuje się operatory genetyczne (krzyżowanie i mutację) zgodnie z ustalonym prawdopodobieństwem. Kolejna różnica między strategiami ewolucyjnymi a algorytmami genetycznymi polega na tym, że parametry algorytmów genetycznych, takie jak prawdopodobieństwo krzyżowania i prawdopodobieństwo mutacji, pozostają stałe w czasie procesu ewolucji, podczas gdy w strategiach ewolucyjnych parametry ulegają ciągłej zmianie (samoadaptacja parametrów).

Strategie ewolucyjne można podzielić na kilka podstawowych typów. Postaramy się poniżej przybliżyć trzy strategie.

7.3.2.1. Strategia ewolucyjna (1 + 1)

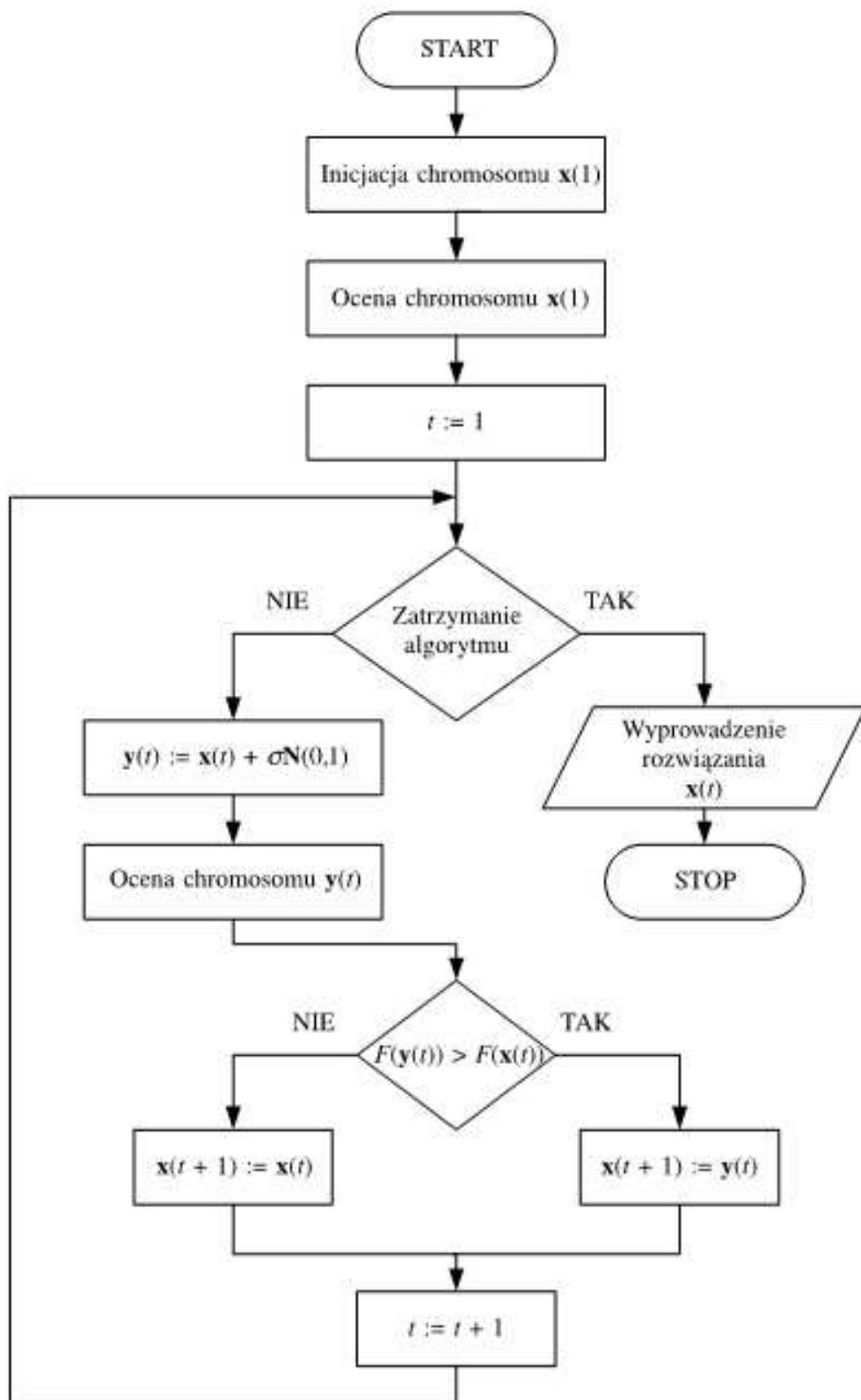
Pierwszym algorytmem, który dał początek całej rodzinie strategii ewolucyjnych, jest strategia (1 + 1). Ogólny schemat strategii (1 + 1) przedstawiono na rysunku 7.7.

W strategii ewolucyjnej (1 + 1) przetwarzany jest jeden chromosom bazowy \mathbf{x} . Algorytm rozpoczyna się od losowego ustalenia wartości poszczególnych składowych wektora \mathbf{x} . W każdej generacji w wyniku mutacji powstaje nowy osobnik \mathbf{y} . Porównując wartości funkcji przystosowania obydwu osobników, tzn. $F(\mathbf{x})$ i $F(\mathbf{y})$, wybierany jest ten z większą wartością funkcji przystosowania (w przypadku poszukiwania maksimum). To właśnie on staje się w następnej generacji nowym chromosomem bazowym \mathbf{x} . W algorytmie tym nie występuje operator krzyżowania. Chromosom \mathbf{y} powstaje poprzez dodanie do każdego z genów chromosomu \mathbf{x} pewnej liczby losowej, generowanej zgodnie z rozkładem normalnym, tzn.

$$y_i = x_i + \sigma N_i(0, 1), \quad (7.17)$$

gdzie: y_i oznacza i -ty gen chromosomu \mathbf{y} , x_i — i -ty gen chromosomu \mathbf{x} , σ — parametr określający zasięg mutacji, $N_i(0, 1)$ — liczbę losową generowaną zgodnie z rozkładem normalnym dla i -tego genu.

Adaptacja parametrów w tej strategii odbywa się poprzez zmianę zasięgu mutacji, tzn. parametru σ . Stosuje się w tym celu regułę 1/5 sukcesów. W myśl tej reguły najlepsze rezultaty w poszukiwaniu optymalnego rozwiązania uzyskuje się, gdy relacja między udanymi a wszystkimi mutacjami wynosi dokładnie 1/5. Gdy przez kolejnych k generacji stosunek udanych mutacji do wszystkich mutacji przewyższa wartość 1/5, wów-



Rys. 7.7. Schemat blokowy strategii ewolucyjnej (1 + 1)

czas zwiększamy wartość parametru σ . Gdy stosunek ten jest mniejszy od 1/5, wtedy zmniejszamy zasięg mutacji. Parametr σ pozostaje niezmieniony, gdy relacja między udanymi a wszystkimi mutacjami wynosi dokładnie 1/5. Reguła 1/5 sukcesów pojawiła się w literaturze jako wynik procesu optymalizacji szybkości zbieżności pewnych wielowymiarowych funkcji testowych, tzw. modelu korytarzowego i sferycznego. Niech $\varphi(k)$ będzie współczynnikiem sukcesów operatora mutacji w poprzednich k generacjach. Regułę 1/5 sukcesów można formalnie zapisać w sposób następujący:

$$\sigma' = \begin{cases} c_1 \cdot \sigma & \text{dla } \varphi(k) < 1/5, \\ c_2 \cdot \sigma & \text{dla } \varphi(k) > 1/5, \\ \sigma & \text{dla } \varphi(k) = 1/5, \end{cases}$$

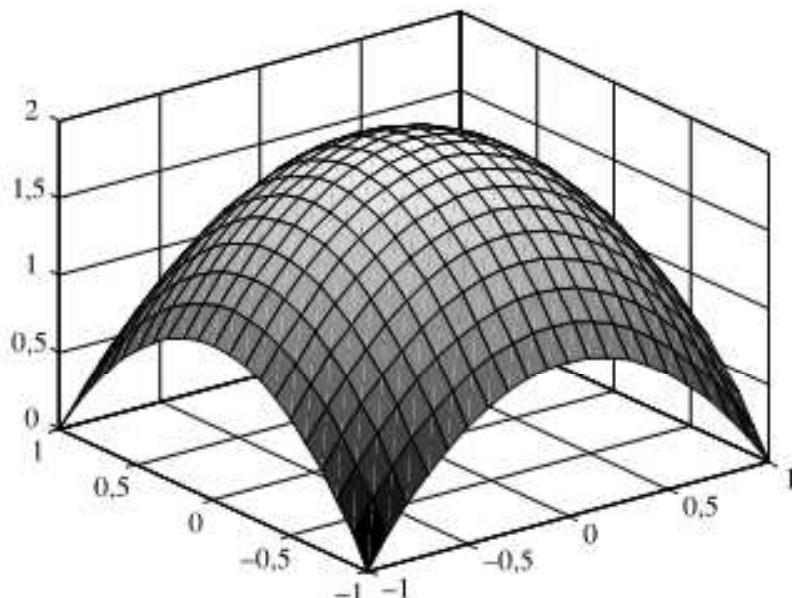
gdzie współczynniki c_1 i c_2 regulują szybkość wzrostu lub zmniejszania się zasięgu mutacji σ . W literaturze dobierano eksperymentalnie wartości tych współczynników: $c_1 = 0,82$, $c_2 = 1/c_1 = 1,2$.

Przykład 7.11

Przeanalizujmy teraz działanie jednej generacji strategii ewolucyjnej (1 + 1) w celu znalezienia maksimum funkcji

$$f(x_1, x_2) = -x_1^2 - x_2^2 + 2 \quad (7.18)$$

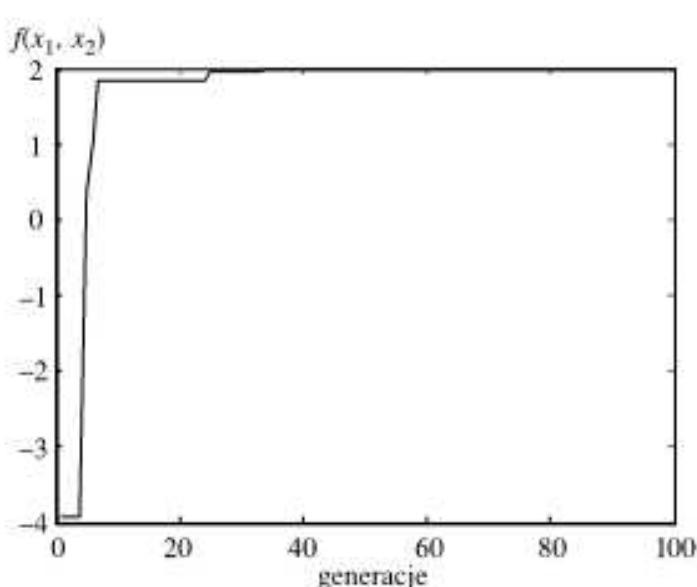
dla $-2 \leq x_1 \leq 2$ i $-2 \leq x_2 \leq 2$. Kształt tej funkcji przedstawia rysunek 7.8. Jest to funkcja dwóch zmiennych i z tyłu też genów składa się chromosom \mathbf{x} . Początkowa wartość zasięgu mutacji $\sigma = 1$.



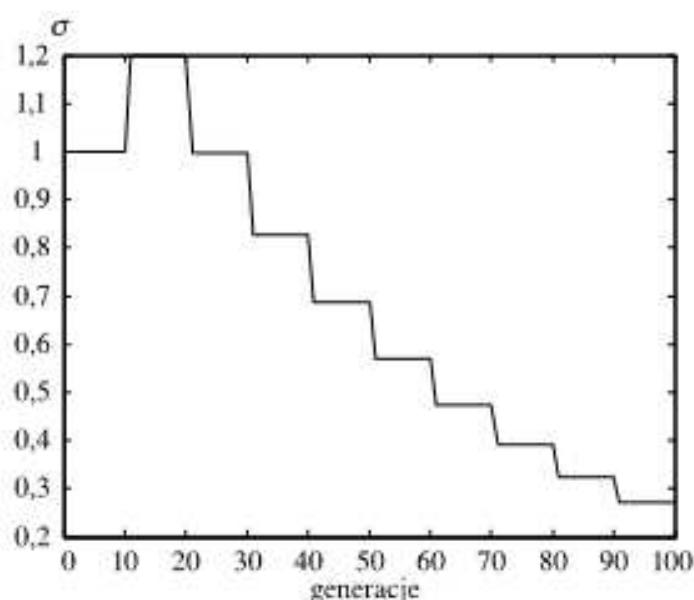
Rys. 7.8. Wykres funkcji $f(x_1, x_2) = -x_1^2 - x_2^2 + 2$ w przykładzie 7.11

Działanie algorytmu rozpoczyna się, zgodnie ze schematem na rysunku 7.7, od wylosowania początkowych wartości składowych wektora \mathbf{x} z przedziału $[-2, 2]$. Do tego przedziału ograniczymy również zakres naszych poszukiwań. Wylosowano wektor liczb $[-1,45, -1,95]$. Oceniamy przystosowanie osobnika za pomocą funkcji przystosowania, którą w naszym przypadku będzie rozpatrywana funkcja (7.18). Otrzymujemy $F(\mathbf{x}) = -3,91$. Rozpoczyna się główna pętla algorytmu. W celu przeprowadzenia operacji mutacji losujemy dwie liczby z rozkładu normalnego $N_1(0, 1) = -0,90$ oraz $N_2(0, 1) = -0,08$. Wykorzystując wzór (7.17), tworzymy nowego osobnika $\mathbf{y} = [-2,35, -2,03]$. Sprawdzamy, czy jego przystosowanie $F(\mathbf{y}) = -7,64$ jest większe od przystosowania osobnika \mathbf{x} . Zauważamy, że w przyszłej generacji ponownie \mathbf{x} będzie chromosomem bazowym. Określmy teraz parametry strategii (1 + 1). Przyjmiemy, że jeżeli reguła 1/5 sukcesów będzie wymagała zwiększenia zasięgu mutacji, to zasięg ten powiększamy, stosując współczynnik $c_2 = 1,2$. W przypadku potrzeby zmniejszenia

zasięgu mutacji, współczynnik ten wynosi $c_1 = 0,82$. Zmiana parametru σ będzie realizowana co $k = 5$ generacji. Zmiany wartości przystosowania chromosomu bazowego w ciągu 100 generacji przedstawia rysunek 7.9, natomiast zmiany wartości zakresu mutacji uwidacznia rysunek 7.10.



Rys. 7.9. Wykres wartości funkcji przystosowania dla 100 generacji w przykładzie 7.11



Rys. 7.10. Wykres zmian zasięgu mutacji σ dla 100 generacji w przykładzie 7.11

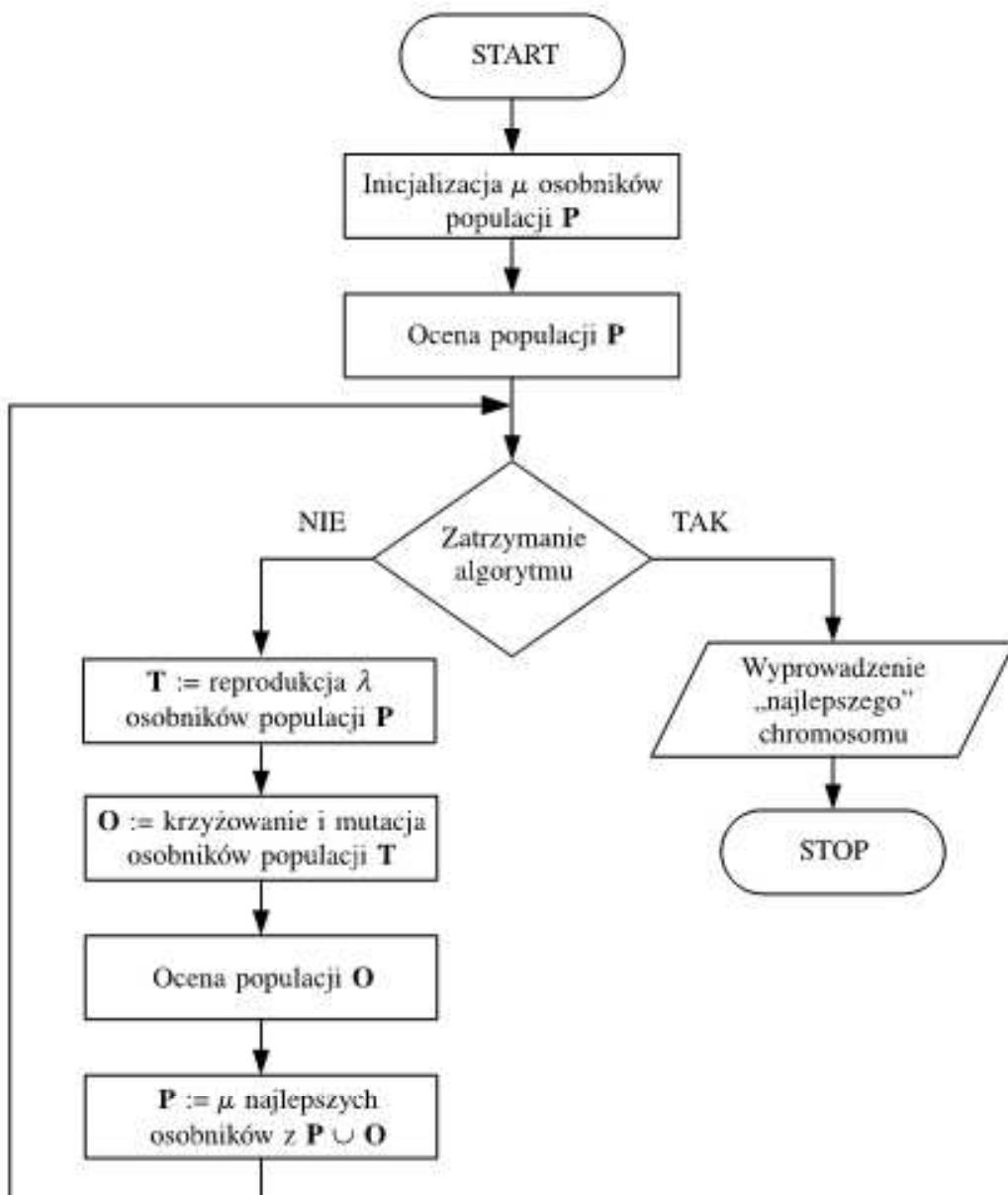
7.3.2.2. Strategia ewolucyjna $(\mu + \lambda)$

Rozwinięciem strategii ewolucyjnej $(1+1)$ jest strategia $(\mu + \lambda)$, której schemat blokowy przedstawiony jest na rysunku 7.11. Algorytm ten, dzięki większej liczbie osobników i w konsekwencji większej różnorodności genotypów, pozwala uniknąć końcowego rozwiązania w postaci minimum lokalnego, często spotykanego w poprzednio opisywanej strategii $(1+1)$.

Algorytm rozpoczynamy od losowego wygenerowania początkowej populacji rodzinieckiej **P**, zawierającej μ osobników. Następnie tworzona jest, poprzez reprodukcję, populacja tymczasowa **T** zawierająca λ osobników, przy czym $\lambda \geq \mu$. Reprodukcja polega na wielokrotnym losowym wyborze λ osobników z populacji **P** (losowanie ze zwracaniem) i umieszczeniu „wybrańców” w populacji tymczasowej **T**. Osobnicy populacji **T** podlegają operacjom krzyżowania i mutacji, w wyniku czego powstaje populacja potomna **O**, również o liczności λ . Ostatnim krokiem jest wybór μ najlepszych potomków z obydwu populacji **P** \cup **O**, które będą stanowiły nową populację rodziniecką **P**.

Strategia $(\mu + \lambda)$ jest rozwinięta o samoczynną adaptację zasięgu mutacji, która zastępuje poprzednio opisaną metodę 1/5 sukcesów. Na potrzeby samoadaptacji dodano do opisu każdego osobnika chromosom σ , zawierający wartości standardowych odchyleń wykorzystywanych podczas mutacji poszczególnych genów chromosomu **x**. Dodatkowo wprowadzono operator krzyżowania. Ważne jest, że operacjom genetycznym ulegają obydwa chromosomy, zarówno wektor zmiennych niezależnych **x**, jak i wektor odchyleń standardowych σ .

Działanie operatora krzyżowania polega na wylosowaniu dwóch osobników i wymianie bądź uśrednieniu wartości ich genów. Dwa nowe osobniki powstałe w ten sposób

Rys. 7.11. Schemat blokowy strategii ewolucyjnej $(\mu + \lambda)$

zastępują swoich rodziców. Opiszemy krzyżowanie polegające na wymianie wartości genów. Wybierzmy dwa osobniki

$$(\mathbf{x}^1, \boldsymbol{\sigma}^1) = ([x_1^1, \dots, x_n^1]^T, [\sigma_1^1, \dots, \sigma_n^1]^T)$$

oraz

$$(\mathbf{x}^2, \boldsymbol{\sigma}^2) = ([x_1^2, \dots, x_n^2]^T, [\sigma_1^2, \dots, \sigma_n^2]^T).$$

W przypadku wymiany wartości genów nowy potomek przybiera postać

$$(\mathbf{x}', \boldsymbol{\sigma}') = ([x_1^{q_1}, \dots, x_n^{q_n}]^T, [\sigma_1^{q_1}, \dots, \sigma_n^{q_n}]^T), \quad (7.19)$$

gdzie $q_i = 1$ lub $q_i = 2$ (każdy gen pochodzi z pierwszego lub z drugiego wybranego rodzica). W przypadku krzyżowania polegającego na uśrednieniu wartości genów dwóch osobników nowego potomka tworzy się według wzoru

$$(\mathbf{x}', \boldsymbol{\sigma}') = ([(x_1^1 + x_1^2) / 2, \dots, (x_n^1 + x_n^2) / 2]^T, [(\sigma_1^1 + \sigma_1^2) / 2, \dots, (\sigma_n^1 + \sigma_n^2) / 2]^T). \quad (7.20)$$

Uśrednianie można również przeprowadzić ze współczynnikiem a wylosowanym z rozkładu jednostajnego $U(0, 1)$ i wówczas krzyżowanie przebiega w sposób następujący:

$$\begin{aligned} x'_i^1 &= ax_i^1 + (1 - a)x_i^2, \\ x'_i^2 &= ax_i^2 + (1 - a)x_i^1, \\ \sigma'_i^1 &= a\sigma_i^1 + (1 - a)\sigma_i^2, \\ \sigma'_i^2 &= a\sigma_i^2 + (1 - a)\sigma_i^1. \end{aligned} \quad (7.21)$$

Operacja mutacji dokonywana jest na pojedynczym osobniku. Jako pierwszy poddawany jest temu zabiegowi chromosom $\sigma = [\sigma_1, \dots, \sigma_n]^T$ zgodnie z formułą

$$\sigma'_i = \sigma_i \exp(\tau' N(0, 1) + \tau N_i(0, 1)), \quad (7.22)$$

w której $i = 1, \dots, n$, n jest długością chromosomu, $N(0, 1)$ — liczbą losową z rozkładu normalnego (losowaną raz dla całego chromosomu), $N_i(0, 1)$ — liczbą losową z rozkładu normalnego (losowaną osobno dla każdego genu), τ' i τ to parametry strategii ewolucyjnej mające wpływ na zbieżność algorytmu. W literaturze spotyka się postać:

$$\tau' = \frac{C}{\sqrt{2n}}, \quad \tau = \frac{C}{\sqrt{2\sqrt{n}}}, \quad (7.23)$$

gdzie C najczęściej przyjmuje wartość 1. Nowe zakresy mutacji σ'_i wpływają na zmianę wartości x_i w sposób następujący:

$$x'_i = x_i + \sigma'_i N_i(0, 1), \quad (7.24)$$

gdzie $N_i(0, 1)$ jest liczbą losową z rozkładu normalnego, $i = 1, \dots, n$.

Jak łatwo zauważyc, wprowadzenie parametru σ'_i danego wzorem (7.22) pozwala na samoadaptację procesu mutacji.

Przykład 7.12

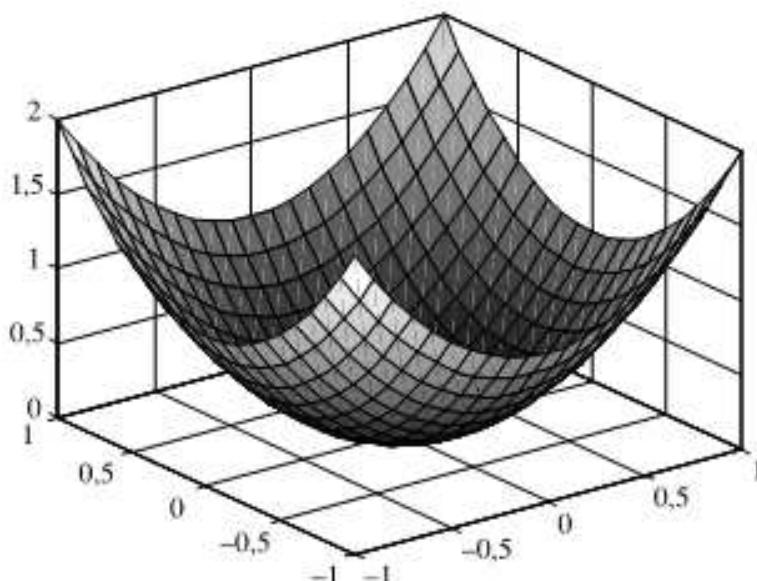
Przedstawimy działanie strategii ewolucyjnej $(\mu + \lambda)$, śledząc kilka jej kroków. Zadanie, na którego przykładzie poznamy ten algorytm, to minimalizacja funkcji

$$f(x_1, x_2) = x_1^2 + x_2^2 \quad (7.25)$$

przy ograniczeniach $-1 \leq x_1 \leq 1$ i $-1 \leq x_2 \leq 1$. Kształt tej funkcji przedstawia rysunek 7.12. Minimum globalne znajduje się w punkcie $(0, 0)$, przy czym $f(0, 0) = 0$. Parametry algorytmu przyjmiemy następujące: $\mu = 4$ oraz $\lambda = 4$. W ramach operacji genetycznych będziemy stosować jedynie operator mutacji. Przystosowanie poszczególnych osobników określa funkcja (7.25), a za najlepiej przystosowane osobniki będziemy uważać te, dla których wartość tej funkcji będzie najmniejsza.

Populacja rodzicielska \mathbf{P} składa się z czterech osobników ($\mu = 4$), a każdy z nich zawiera dwuelementowe wektory $\mathbf{x} = [x_1, x_2]^T$ oraz $\sigma = [\sigma_1, \sigma_2]^T$. Początkową populację rodzicielską generujemy losowo. Wartości x_1 oraz x_2 losujemy z zakresu $[-1, 1]$ oraz przyjmujemy $\sigma_1 = \sigma_2 = 1$. Tabela 7.1 przedstawia chromosomy początkowej populacji \mathbf{P} oraz wartości ich funkcji przystosowania.

Łatwo zauważyc, że chromosomy numer 1 i 4 charakteryzują się najmniejszą wartością funkcji przystosowania. Tworzymy teraz tymczasową populację \mathbf{T} o liczności $\lambda = 4$.



Rys. 7.12. Wykres funkcji dwuwymiarowej w przykładzie 7.12

Tabela 7.1. Populacja rodzicielska **P**

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	0,63	0,41	1	1	0,57
2	0,57	-0,91	1	1	1,15
3	-0,67	-0,62	1	1	0,83
4	0,38	0,65	1	1	0,57

Za pomocą losowania ze zwracaniem, z populacji rodzicielskiej **P** wybieramy osobniki o numerach 4, 2, 3 i 4. Tymczasową populację **T** przedstawia tabela 7.2. Zauważmy, że do populacji **T** został wybrany tylko jeden najlepszy chromosom z populacji **P**.

Tabela 7.2. Populacja tymczasowa **T**

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	0,38	0,65	1	1	0,57
2	0,57	-0,91	1	1	1,15
3	-0,67	-0,62	1	1	0,83
4	0,38	0,65	1	1	0,57

Kolejnym krokiem są operacje genetyczne dokonywane na osobnikach tymczasowej populacji, czego wynikiem będzie populacja potomna **O**. Mutacja chromosomu σ wymaga ustalenia parametrów τ' oraz τ zgodnie ze wzorem (7.23). Przyjmujemy, że $C = 1$. Wówczas dla $n = 2$ parametry τ' i τ są równe, odpowiednio, 0,5 i 0,5946. Przebieg mutacji elementów σ_i przedstawiono w tabeli 7.3, natomiast elementów x_i w tabeli 7.4.

Tabela 7.3. Przebieg mutacji chromosomu σ poszczególnych osobników populacji **T**

Nr osobnika	$N(0,1)$	Gen 1				Gen 2			
		σ_1	$N_1(0,1)$	$\exp(\tau' N(0,1) + \tau N_1(0,1))$	σ'_1	σ_2	$N_2(0,1)$	$\exp(\tau' N(0,1) + \tau N_2(0,1))$	σ'_2
1	1,27	1	0,47	2,50	2,50	1	-0,38	1,51	1,51
2	-0,58	1	0,05	0,77	0,77	1	-0,46	0,57	0,57
3	0,47	1	-0,82	0,78	0,78	1	0,44	1,64	1,64
4	-2,38	1	0,31	0,37	0,37	1	-1,05	0,16	0,16

Tabela 7.4. Przebieg mutacji chromosomu x poszczególnych osobników populacji **T**

Nr osobnika	Gen 1				Gen 2			
	x_1	$N_1(0,1)$	$\sigma_1 N_1(0,1)$	x'_1	x_1	$N_2(0,1)$	$\sigma_2 N_2(0,1)$	x_2
1	0,38	-0,27	-0,67	-0,29	0,65	-1,03	-1,55	-0,90
2	0,57	0,20	0,15	0,72	-0,91	-0,30	-0,17	-1,08
3	-0,67	-1,14	-0,89	-1,56	-0,62	-1,32	-2,17	-2,79
4	0,38	-0,27	-0,10	0,28	0,65	-1,71	-0,28	0,37

Po przeprowadzeniu operacji genetycznych ostatecznie uzyskujemy populację **O** przedstawioną w tabeli 7.5.

Tabela 7.5. Populacja potomna **O**

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	-0,29	-0,90	2,50	1,51	0,89
2	0,72	-1,08	0,77	0,57	1,68
3	-1,56	-2,79	0,78	1,64	10,22
4	0,28	0,37	0,37	0,16	0,22

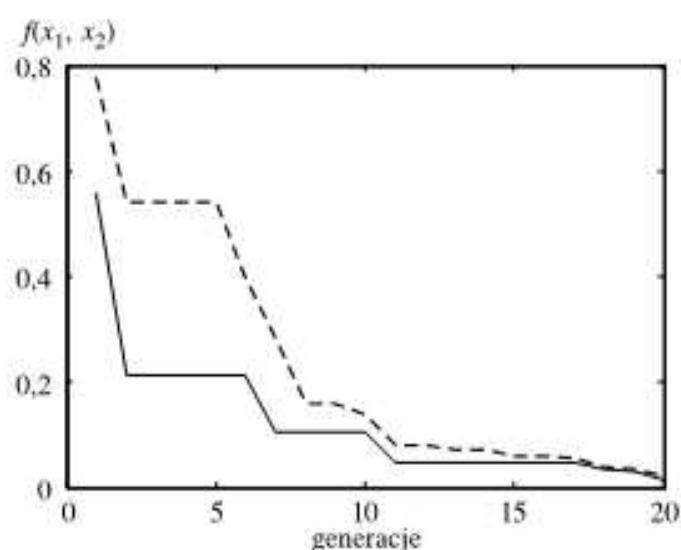
Zgodnie z zasadą działania strategii $(\mu + \lambda)$, nową pulę rodzicielską **P** tworzy μ najlepszych chromosomów ze starej populacji **P** (tab. 7.1) oraz populacji **O** (tab. 7.5). Wybieramy więc osobnika o numerze 4 z populacji **O** oraz osobniki o numerach 1, 4 i 3 z populacji **P**. Nową populację przedstawia tabela 7.6.

Tabela 7.6. Nowa populacja rodzicielska **P**

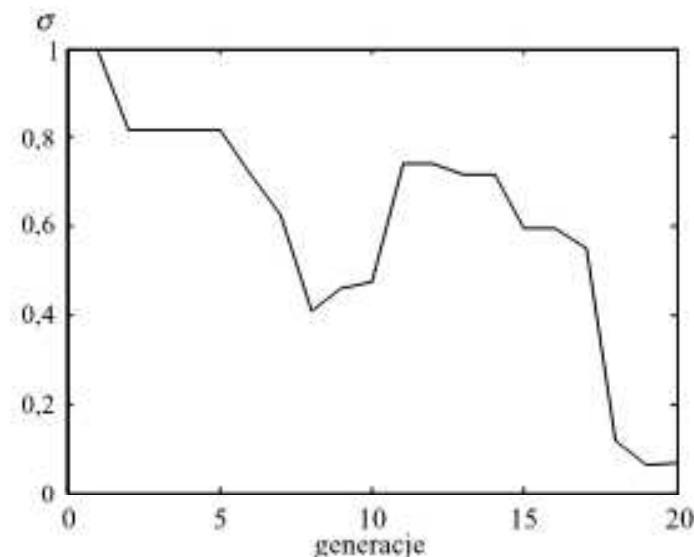
Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	0,28	0,37	0,37	0,16	0,22
2	0,63	0,41	1	1	0,57
3	0,38	0,65	1	1	0,57
4	-0,67	-0,62	1	1	0,83

Zwrócić uwagę na najlepszego osobnika w nowej populacji. Powstał on w wyniku przeprowadzenia operacji genetycznych i charakteryzuje się wartościami x_1 i x_2 bliskimi rozwiązaniu optymalnego (minimum funkcji). Zauważmy, że wartości elementów σ_1 i σ_2 odpowiadających najlepszemu osobnikowi są wyraźnie mniejsze od początkowo przyjętych. Ten mały zakres mutacji umożliwił uzyskanie dokładniejszego rozwiązania, a także, co najważniejsze, zostanie przekazany nowej populacji, umożliwiając zawężenie przeszukiwanej przestrzeni. Zauważony przez nas wcześniej osobnik numer 1 ze starej populacji \mathbf{P} ciągle jest jednym z lepszych osobników i znalazł swoje miejsce w nowej puli rodzicielskiej.

Na rysunku 7.13 przedstawiono najlepsze (linia ciągła) i średnie (linia przerywana) wartości funkcji przystosowania osobników w kolejnych 20 generacjach strategii ewolucyjnej ($\mu + \lambda$). Widać wyraźną zbieżność do minimum funkcji (7.25).



Rys. 7.13. Wykres najlepszej (linia ciągła) i średniej (linia przerywana) wartości funkcji przystosowania w kolejnych generacjach algorytmu w przykładzie 7.12



Rys. 7.14. Wykres średniej wartości zasięgu mutacji w całej populacji w kolejnych generacjach algorytmu w przykładzie 7.12

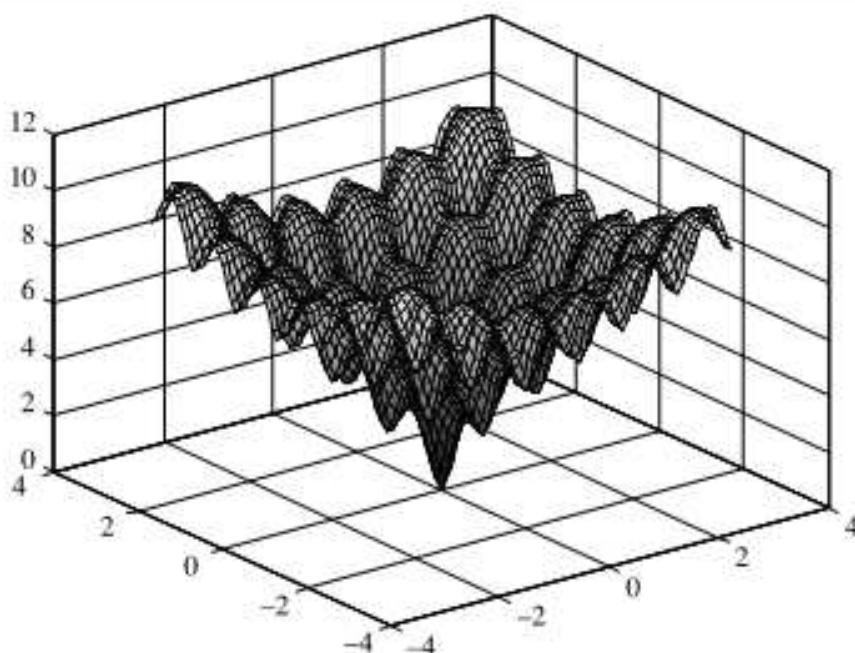
Na rysunku 7.14 pokazano średnie wartości zasięgu mutacji w 20 kolejnych generacjach. Wartości te, mimo pewnych wahań, mają tendencję malejącą.

Przykład 7.13

Kolejnym przykładem wykorzystania strategii ewolucyjnej ($\mu + \lambda$) będzie poszukiwanie minimum funkcji testowej Ackleya. Funkcje testowe wykorzystywane są do sprawdzenia poprawności i efektywności działania algorytmów ewolucyjnych. *Funkcja Ackleya* wyrażona jest wzorem

$$f(\mathbf{x}) = -20 \exp \left(-0,2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e \quad (7.26)$$

przy ograniczeniach $-30 \leq x_i \leq 30$, przy czym n jest liczbą zmiennych. Istnieje jedno minimum globalne tej funkcji $\mathbf{x} = 0$ i wówczas $f(\mathbf{x}) = 0$. Dla dwóch zmiennych x_1 i x_2 , fragment wykresu funkcji przedstawia rysunek 7.15.

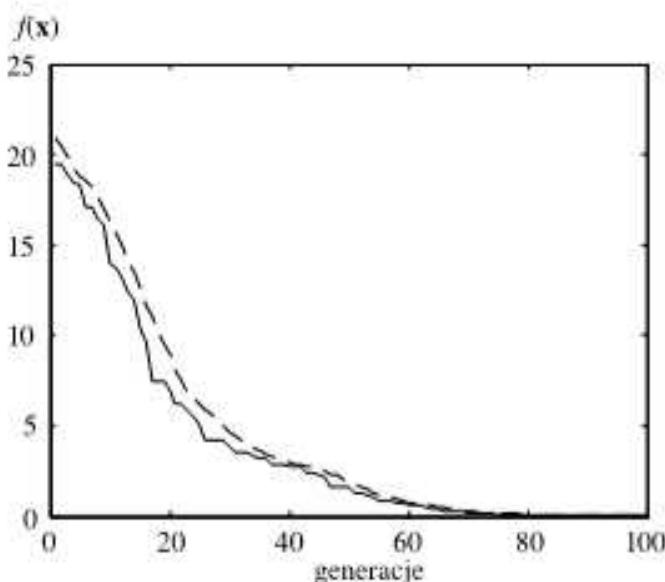


Rys. 7.15. Wykres funkcji (7.26) dla $n = 2$

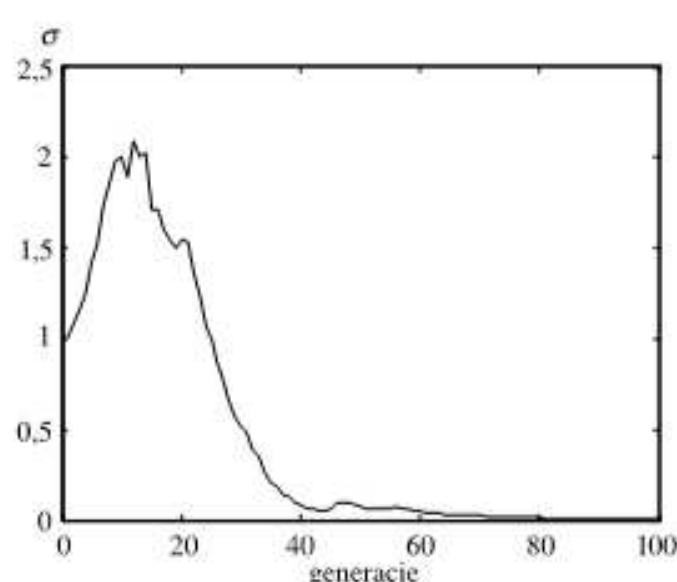
W naszym zadaniu poszukujemy minimum 10-wymiarowej funkcji Ackleya. Pojedynczy osobnik zatem składa się z 10-elementowych wektorów \mathbf{x} oraz σ . Parametry strategii $(\mu + \lambda)$ przyjmiemy dla eksperymentu następujące: $\mu = 50$ oraz $\lambda = 200$. Liczba generacji wynosi 100. Elementy wektorów \mathbf{x} poszczególnych chromosomów zostaną zainicjowane wartościami losowymi z zakresu $[-30, 30]$, natomiast wszystkie zakresy mutacji, tzn. elementy wektorów σ początkowo przyjmują wartość równą 1. Zgodnie ze wzorem (7.23) parametry τ' i τ dla $C = 1$ i $n = 10$ wyznaczamy następująco:

$$\tau' = \frac{1}{\sqrt{20}} = 0,2236, \quad \tau = \frac{1}{\sqrt{2\sqrt{10}}} = 0,3976. \quad (7.27)$$

W przykładzie nie wykorzystujemy operatora krzyżowania. Na rysunku 7.16 można



Rys. 7.16. Wykres najlepszej (linia ciągła) i średniej (linia przerywana) wartości funkcji przystosowania w kolejnych generacjach algorytmu w przykładzie 7.13

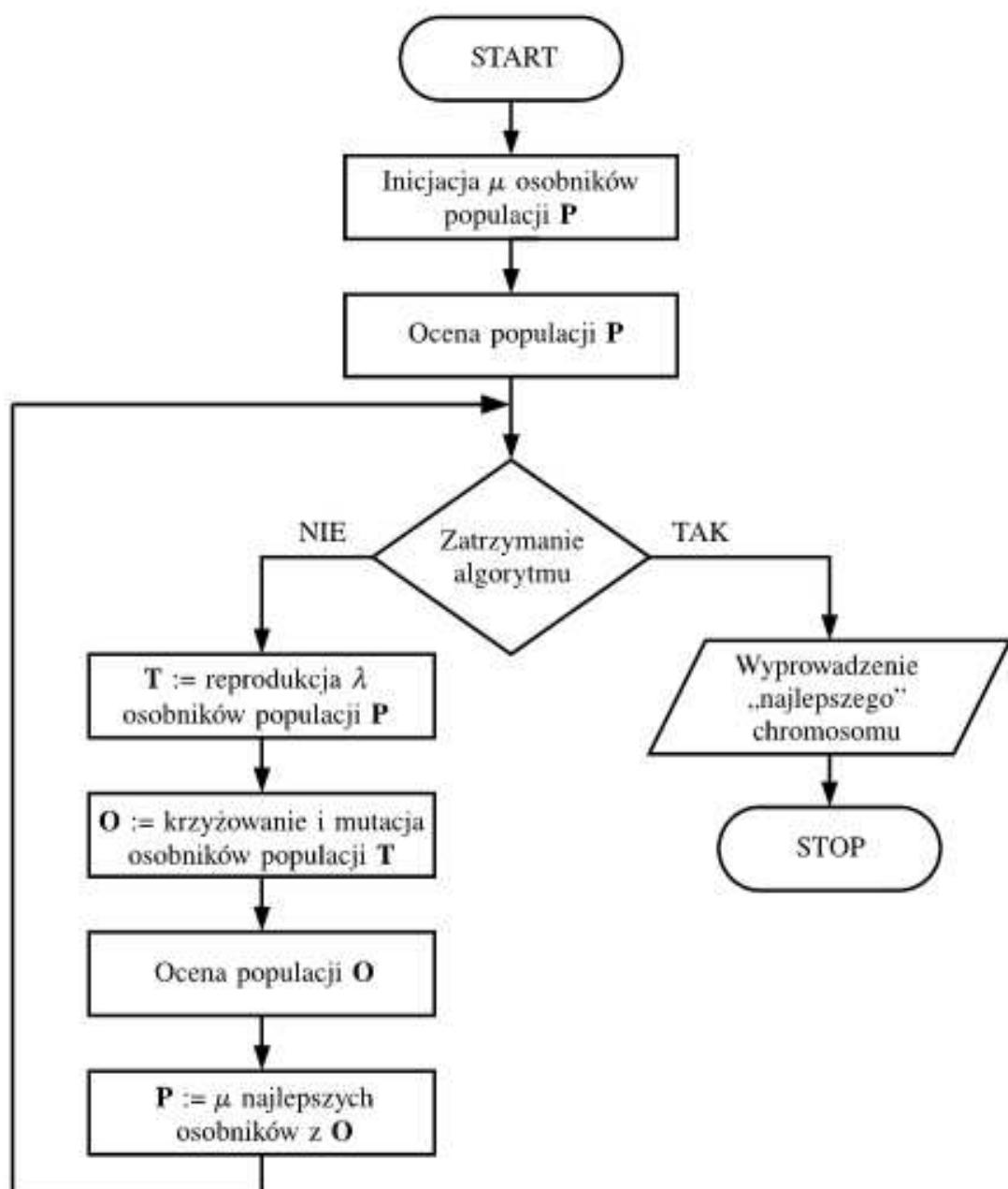


Rys. 7.17. Wykres średniej wartości zasięgu mutacji w całej populacji w kolejnych generacjach algorytmu w przykładzie 7.13

zaobserwować przebieg działania 100 generacji strategii ewolucyjnej ($\mu + \lambda$). Około 70. generacji najlepsze wartości funkcji przystosowania zbliżają się do minimum globalnego. Porównajmy to ze zmianą średniej wartości zasięgu mutacji (rys. 7.17). Początkowy jej wzrost można odczytać jako zwiększenie różnorodności populacji, a tym samym poszerzenie zakresu poszukiwań. Zaraz potem następuje gwałtowny spadek i zmniejszają się różnice w chromosomach wywoływanie operatorem genetycznym. Osobniki zaczynają oscylować wokół ostatecznego rozwiązania.

7.3.2.3. Strategia ewolucyjna (μ, λ)

Częściej od strategii ($\mu + \lambda$) stosowana jest strategia (μ, λ). Działanie obydwu algorytmów jest prawie identyczne. Różnica polega na tym, że nową populację **P** zawierającą μ osobników wybiera się tylko spośród najlepszych λ osobników populacji **O**. Aby było to możliwe, musi być spełniony warunek $\mu > \lambda$. Schemat blokowy przedstawiający działanie algorytmu przedstawiono na rysunku 7.18.



Rys. 7.18. Schemat blokowy strategii ewolucyjnej (μ, λ)

Metoda ta ma przewagę nad strategią $(\mu + \lambda)$ w jednym dość ważnym punkcie, a mianowicie populacja do tej pory mogła być zdominowana jednym osobnikiem o dużej wartości funkcji przystosowania, ale zbyt dużych lub zbyt małych wartościach odchylen standardowych. Utrudniałoby to znalezienie lepszych rozwiązań. Strategia (μ, λ) pozbawiona jest tej wady, ponieważ stare osobniki nie przechodzą do nowej puli rodzicielskiej. Operatory genetyczne, które można zastosować, nie różnią się od już poznanego krzyżowania (7.19)–(7.21) oraz mutacji (7.22) i (7.24).

Przykład 7.14

Prześledźmy kilka kroków działania strategii ewolucyjnej (μ, λ) , rozwiązuając proste zadanie znalezienia maksimum funkcji

$$f(x_1, x_2) = -x_1^2 - x_2^2, \quad (7.28)$$

przy czym $-1 \leq x_1 \leq 1$ i $-1 \leq x_2 \leq 1$. Podobnie jak w przykładzie 7.12, pojedynczy osobnik będzie składał się z dwuelementowych wektorów \mathbf{x} oraz σ . Przyjmujemy następujące parametry: $\mu = 4$, $\lambda = 8$. Algorytm strategii ewolucyjnej rozpoczynamy od wygenerowania początkowych wartości chromosomów populacji rodzicielskiej. Składowe wektora \mathbf{x} zainicjujemy liczbami losowymi z zakresu poszukiwanego rozwiązania, czyli $[-1, 1]$, natomiast elementy wektora σ zainicjujemy wartościami równymi 1. W tabeli 7.7 przedstawiono początkowe wartości składowych wektorów \mathbf{x} i σ czterech wylosowanych osobników.

Tabela 7.7. Początkowa populacja rodzicielska \mathbf{P}

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	-0,67	-0,68	1	1	-0,91
2	0,36	-1,00	1	1	-1,13
3	-0,97	-0,83	1	1	-1,63
4	-0,19	0,98	1	1	-1,00

Za najlepiej przystosowane chromosomy w populacji będziemy uważać te, które charakteryzują się największą wartością funkcji (7.28). Wybieramy teraz drogą losowania ze zwracaniem 8 osobników do tymczasowej populacji \mathbf{T} . Założymy, że wylosowano chromosomy o następujących numerach: 2, 2, 2, 3, 3, 1, 3, 4. Populację \mathbf{T} pokazuje tabela 7.8.

Tabela 7.8. Populacja tymczasowa \mathbf{T}

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	0,36	-1,00	1	1	-1,13
2	0,36	-1,00	1	1	-1,13
3	0,36	-1,00	1	1	-1,13
4	-0,97	-0,83	1	1	-1,63
5	-0,97	-0,83	1	1	-1,63
6	-0,67	-0,68	1	1	-0,91
7	-0,97	-0,83	1	1	-1,63
8	-0,19	0,98	1	1	-1,00

Podobnie jak w przykładzie 7.12, dla $C = 1$ i $n = 2$, $\tau' = 0,5$ i $\tau = 0,5946$. Przebieg operacji mutacji dla poszczególnych składowych wektorów σ pokazano w tabeli 7.9.

Tabela 7.9. Przebieg mutacji genów chromosomu σ

Nr osobnika	$N(0, 1)$	Gen 1				Gen 2			
		σ_1	$N_1(0, 1)$	$\exp(\tau'N(0, 1) + \tau N_1(0, 1))$	σ'_1	σ_2	$N_2(0, 1)$	$\exp(\tau'N(0, 1) + \tau N_2(0, 1))$	σ'_2
1	-0,94	1	1,16	1,25	1,25	1	-0,04	0,61	0,61
2	0,15	1	-0,67	0,72	0,72	1	-1,00	0,59	0,59
3	0,80	1	-1,74	0,53	0,53	1	-0,42	1,16	1,16
4	-1,44	1	-1,01	0,27	0,27	1	-1,37	0,22	0,22
5	-0,77	1	0,05	0,70	0,70	1	-0,36	0,55	0,55
6	-1,67	1	0,27	0,51	0,51	1	0,80	0,70	0,70
7	-0,11	1	1,31	2,06	2,06	1	-0,03	0,93	0,93
8	-2,36	1	0,71	0,47	0,47	1	0,49	0,41	0,41

Druga kolumna tabeli 7.9 zawiera wartości liczb (zmienne losowe o rozkładzie normalnym) wylosowanych dla całego chromosomu. Natomiast w pozostałych kolumnach przedstawiono, kolejno, wartości parametru σ_i przed mutacją, wartości wylosowane $N_i(0, 1)$ dla danego genu, współczynnik $\exp(\tau'N(0, 1) + \tau N_i(0, 1))$, przez który zostanie pomnożona poprzednia wartość σ_i , oraz nową już wartość zakresu mutacji dla obu genów chromosomu $\sigma = [\sigma_1, \sigma_2]$. W tabeli 7.10 pokazano przebieg mutacji chromosomów \mathbf{x} z wykorzystaniem nowych wartości zakresu mutacji σ' .

Tabela 7.10. Przebieg mutacji genów chromosomu \mathbf{x}

Nr osobnika	Gen 1				Gen 2			
	x_1	$N_1(0, 1)$	$\sigma_1 N_1(0, 1)$	x'_1	x_1	$N_2(0, 1)$	$\sigma_2 N_2(0, 1)$	x'_2
1	0,36	-0,50	-0,62	-0,26	-1,00	-0,05	-0,03	-1,03
2	0,36	1,04	0,75	1,11	-1,00	-0,70	-0,42	-1,42
3	0,36	0,78	0,41	0,77	-1,00	1,90	2,21	1,21
4	-0,97	1,23	0,33	-0,64	-0,83	-1,54	-0,33	-1,16
5	-0,97	-0,48	-0,34	-1,31	-0,83	-0,70	-0,38	-1,21
6	-0,67	0,84	0,43	-0,24	-0,68	0,44	0,31	-0,37
7	-0,97	-0,42	-0,87	-1,84	-0,83	0,51	0,47	-0,36
8	-0,19	0,81	0,38	0,19	0,98	-0,12	-0,05	0,93

Kolejne kolumny tabeli 7.10 przedstawiają wartości genów x_i przed mutacją, wylosowane liczby $N_i(0, 1)$, współczynniki, o jakie zmienione zostaną geny x_i , oraz nowe wartości genów x'_i . Populację potomną **O** przedstawia tabela 7.11.

Nową populację rodzicielską tworzą cztery najlepsze osobniki z populacji **O**. W naszym przypadku mają one numery 6, 8, 1 i 4. Osobniki nowej populacji **P** pokazuje tabela 7.12.

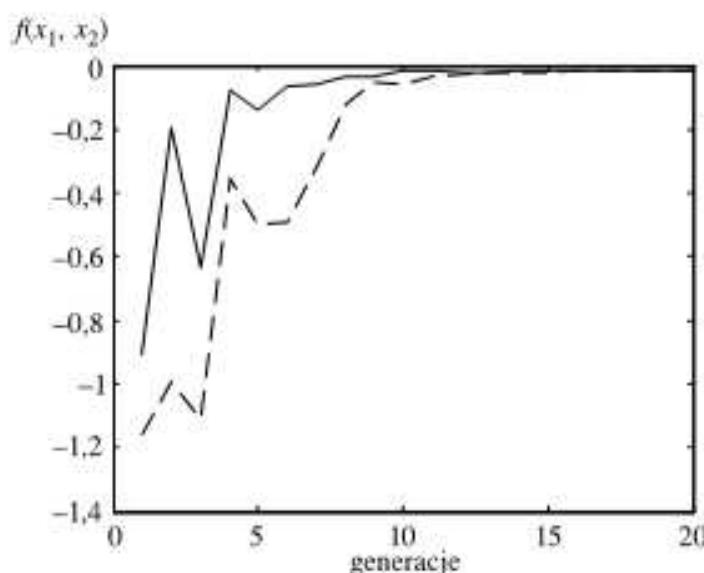
Tabela 7.11. Populacja potomna **O**

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	-0,26	-1,03	1,25	0,61	-1,13
2	1,11	-1,42	0,72	0,59	-3,25
3	0,77	1,21	0,53	1,16	-2,06
4	-0,64	-1,16	0,27	0,22	-1,76
5	-1,31	-1,21	0,70	0,55	-3,18
6	-0,24	-0,37	0,51	0,70	-0,19
7	-1,84	-0,36	2,06	0,93	-3,52
8	0,19	0,93	0,47	0,41	-0,90

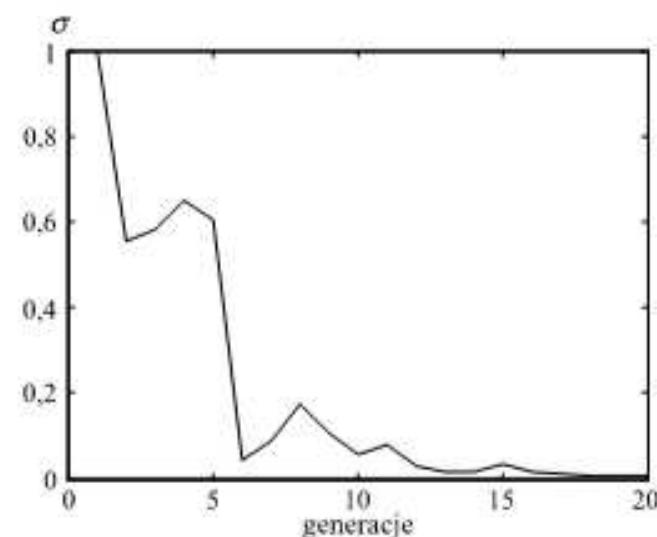
Tabela 7.12. Nowa populacja rodzicielska **P**

Nr osobnika	x_1	x_2	σ_1	σ_2	$f(x_1, x_2)$
1	-0,24	-0,37	0,51	0,70	-0,19
2	0,19	0,93	0,47	0,41	-0,90
3	-0,26	-1,03	1,25	0,61	-1,13
4	-0,64	-1,16	0,27	0,22	-1,76

Porównując tabelle 7.7 i 7.12, widać poprawę rezultatów. Nowa populacja **P** charakteryzuje się zbliżeniem do rozwiązania optymalnego. Warto również zauważać zmniejszenie się zakresów mutacji. Prześledźmy teraz działanie algorytmu, analizując przebieg kolejnych generacji. Rysunek 7.19 przedstawia wartości najlepszej funkcji przystosowania



Rys. 7.19. Wykres najlepszej (linia ciągła) i średniej (linia przerywana) wartości funkcji przystosowania w kolejnych generacjach algorytmu w przykładzie 7.14



Rys. 7.20. Wykres średniej wartości zasięgu mutacji w całej populacji w kolejnych generacjach algorytmu w przykładzie 7.14

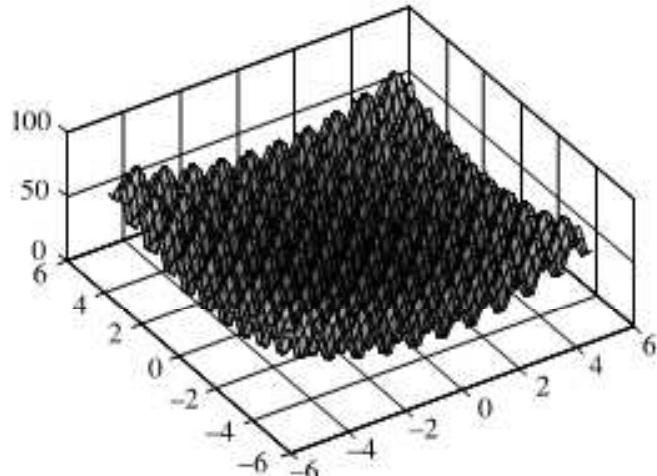
nia (linia ciągła) oraz wartości średniej funkcji przystosowania (linia przerywana) całej populacji. Warto przypomnieć, że najlepiej przystosowane osobniki z poprzedniej populacji nie przechodzą do nowo tworzonych populacji, jak w strategii $(\mu + \lambda)$. Wyraźnie widać na wykresie, że w 2. generacji osiągnięto poprawę wyniku działania algorytmu, ale już w następnym kroku obserwujemy gorsze rezultaty.

Przykład 7.15

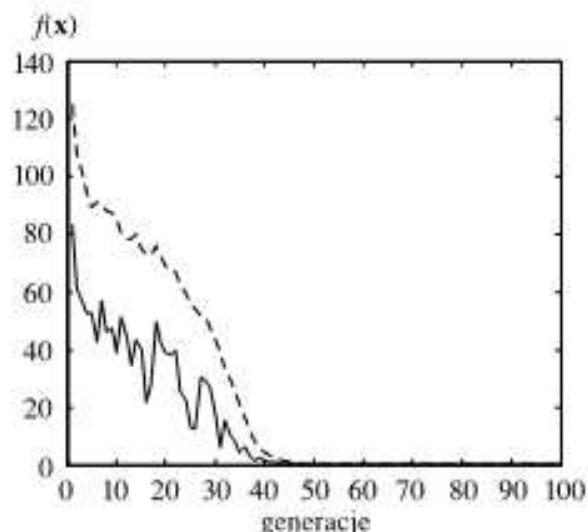
Zastosujemy strategię (μ, λ) do poszukiwania minimum dość skomplikowanej funkcji testowej *Rastingira*, która wyrażona jest wzorem

$$f(\mathbf{x}) = An + \sum_{i=1}^n x_i^2 - A \cos(2\pi x_i). \quad (7.29)$$

Przyjmiemy $A = 10$ oraz $n = 10$, $-5,21 \leq x_i \leq 5,21$, $i = 1, \dots, 10$. Jest to funkcja (rys. 7.21) z siatką minimów lokalnych oraz jednym minimum globalnym w punkcie $\mathbf{x} = \mathbf{0}$, w którym $f(\mathbf{x}) = 0$.

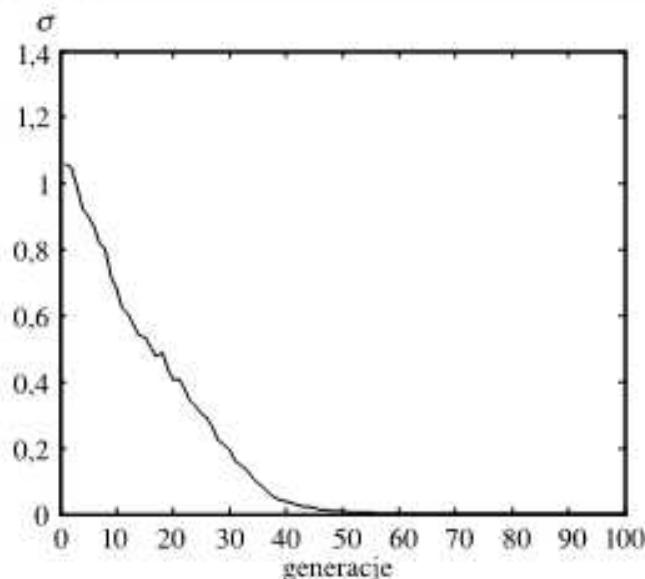


Rys. 7.21. Wykres funkcji dwuwymiarowej Rastingira



Rys. 7.22. Wykres najlepszej (linia ciągła) i średniej (linia przerywana) wartości funkcji przystosowania w kolejnych generacjach algorytmu w przykładzie 7.15

Ustalamy następujące parametry algorytmu: $\mu = 100$ oraz $\lambda = 400$. Liczba generacji wynosi 100. Zastosujemy operator mutacji oraz operator krzyżowania (7.21). Początkowo wylosowane wartości elementów wektora \mathbf{x} wszystkich osobników z puli rodzicielskiej pochodzą z zakresu $[-5,21; 5,21]$. Składowe wektora σ , podobnie jak w poprzednich przykładach, początkowo przyjmują wartość 1. Przebieg zmian wartości najlepszej funkcji przystosowania (linia ciągła) oraz średniej wartości funkcji przystosowania (linia przerywana) całej populacji w kolejnych generacjach algorytmu można prześledzić na rysunku 7.22. Już po około 40. generacji widać, że rozwiązanie zbliżyło się do wartości poszukiwanego minimum. Wykres wartości przystosowania najlepszych osobników jest wyraźnie „poszarpany”. Wynika to z faktu, że w kolejnych krokach traci się informację o poprzednim najlepszym rozwiązaniu. Jak wspomnieliśmy, jest to ważne dla unikania minimów lokalnych. Rysunek 7.23 przedstawia przebieg zmian zasięgu



Rys. 7.23. Wykres średniej wartości zasięgu mutacji populacji w kolejnych generacjach algorytmu w przykładzie 7.15

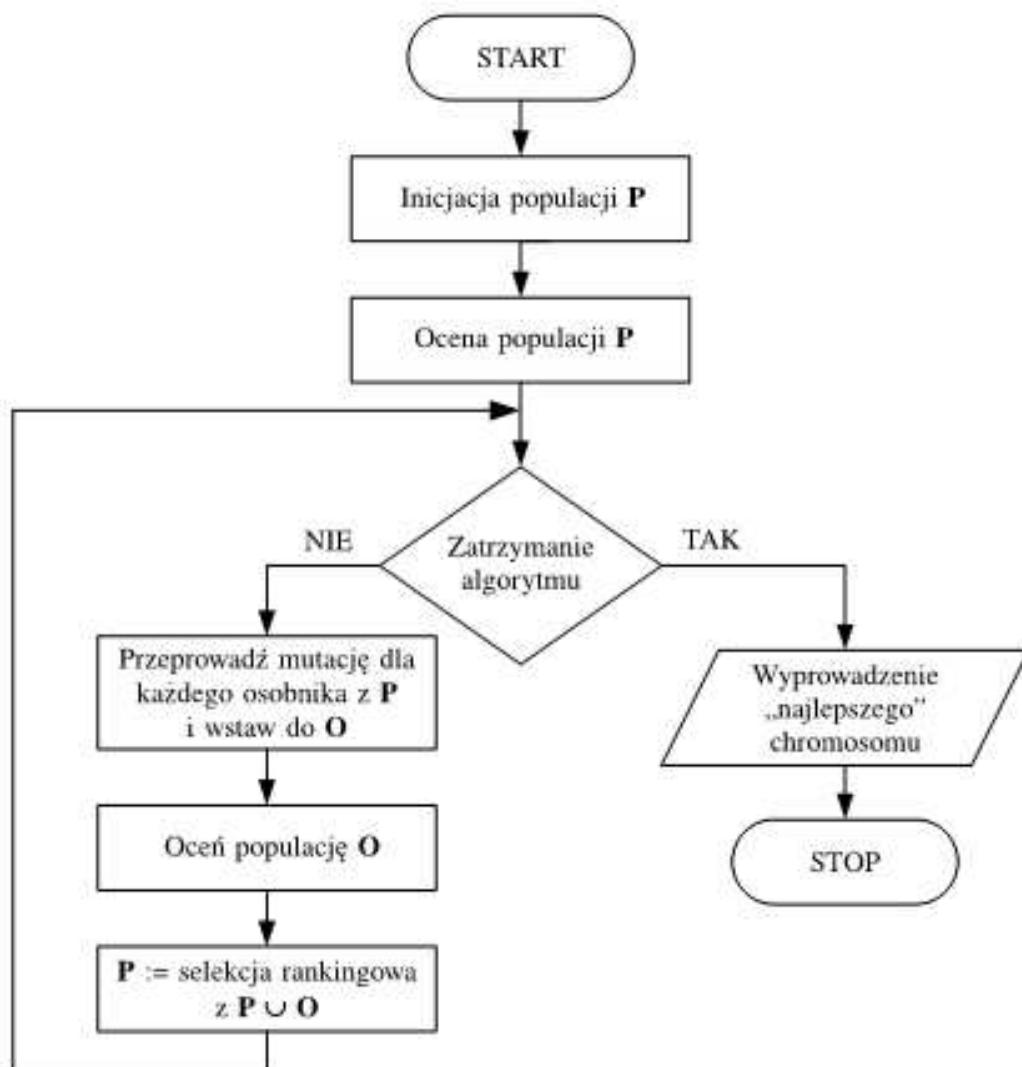
mutacji. Wraz ze zbliżaniem się do rozwiązania optymalnego, średnia wartość zasięgu mutacji zdecydowanie maleje.

7.3.3. Programowanie ewolucyjne

Pierwotnie programowanie ewolucyjne rozwijało się w kontekście odkrywania gramatyki nieznanego języka. Gramatyka była modelowana z wykorzystaniem automatu skończonego, który był poddawany ewolucji. Wyniki okazywały się obiecujące, jednak swoją popularność programowanie ewolucyjne zyskało wówczas, gdy rozwinęło się w kierunku optymalizacji numerycznej. Na rysunku 7.24 przedstawiono schemat blokowy algorytmu programowania ewolucyjnego. Patrząc na ten schemat, można zauważać duże podobieństwo do strategii ewolucyjnej ($\mu + \lambda$). Istnieje jednak dość znacząca różnica. Podczas każdej generacji algorytmu programowania ewolucyjnego nowa populacja **O** jest tworzona poprzez mutację każdego z osobników populacji rodzicielskiej **P**. Natomiast w strategii ewolucyjnej ($\mu + \lambda$) każdy z osobników ma jednakową szansę na pojawienie się w populacji tymczasowej **T**, na której przeprowadzane są operacje genetyczne, przy czym $\lambda \geq \mu$. W programowaniu ewolucyjnym populacje **P** oraz **O** są tak samo liczne, tzn. $\mu = \lambda$. Ostatecznie nowa populacja rodzicielska **P** jest tworzona za pomocą selekcji rankingowej, której podlegają zarówno osobniki ze starej populacji **P**, jak i osobniki zmutowane z populacji **O**. Warto wspomnieć, że mutacje osobników w programowaniu ewolucyjnym polegają na losowej perturbacji wartości poszczególnych genów.

7.3.4. Programowanie genetyczne

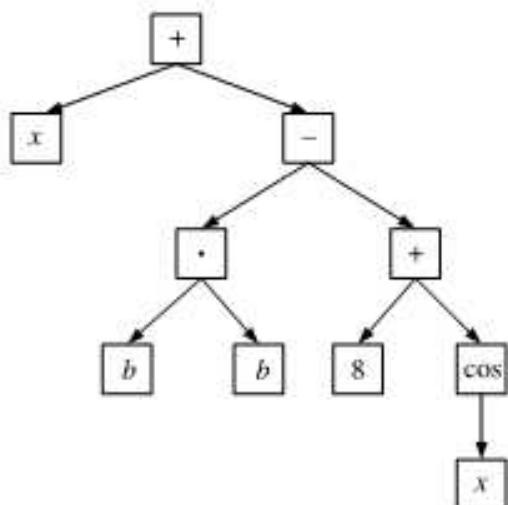
Programowanie genetyczne jest rozszerzeniem klasycznego algorytmu genetycznego i może być wykorzystane do automatycznego generowania programów komputerowych. Programowanie genetyczne stosuje język programowania LISP, w którym program jest reprezentowany tak samo jak dane, tzn. w postaci drzewa. Dlatego też w programowaniu genetycznym kodowanie binarne zostało zastąpione kodowaniem drzewiastym. W węzłach tego drzewa znalazły się wielkości niebinarne, np. wartości liczbowe, zmienne,



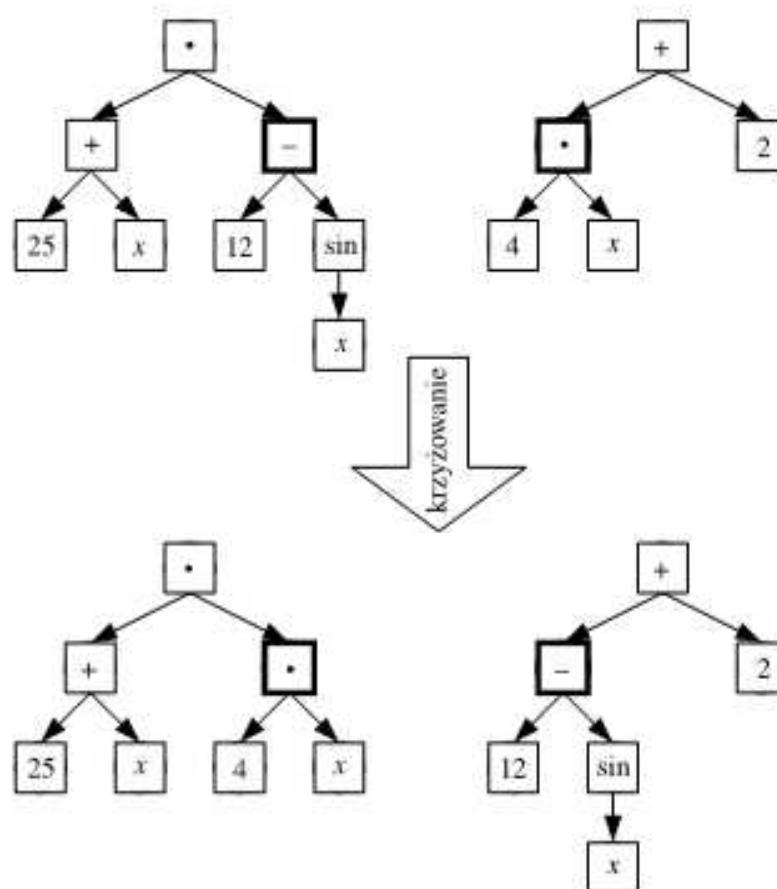
Rys. 7.24. Schemat blokowy algorytmu programowania ewolucyjnego

funkcje lub symbole pewnego alfabetu. Wprowadzenie takiego kodowania pociągnęło też za sobą wprowadzenie nowych operatorów genetycznych. Chromosomy mają specyficzną budowę, więc tradycyjne metody krzyżowania i mutacji, znane z klasycznego algorytmu genetycznego, nie mają tu zastosowania. W programowaniu genetycznym chromosomy są kodowane jako drzewa składające się z węzłów i krawędzi. Właściwa informacja zawarta jest w węzłach. Natomiast krawędzie określają wzajemne relacje pomiędzy węzłami. Rozróżniamy węzły terminalne (nie mające węzłów podrzędnych) oraz węzły pośrednie (przeciwieństwo węzłów terminalnych — mają połączenia z następnymi węzłami). W kodowaniu drzewiastym wyróżniamy jeden węzeł stanowiący korzeń drzewa. Węzeł ten nie ma węzła nadzakaznego. Na rysunku 7.25 pokazano przykład kodowania drzewiastego. Zakodowaną informacją jest funkcja $x + b \cdot b - (8 + \cos(x))$.

Jak już wspomnieliśmy, w przypadku kodowania drzewiastego istnieją specyficzne operatory krzyżowania i mutacji. Krzyżowanie polega na wykorzystaniu dwóch chromosomów rodzicielskich i utworzeniu na ich podstawie dwóch nowych potomków. Losowo wybieramy w każdym z chromosomów przeznaczonych do krzyżowania jeden węzeł. Może to być zarówno węzeł terminalny, jak i pośredni. Węzły te, wraz z odpowiadającymi im poddrzewami, są zamieniane ze sobą miejscami. W wyniku tej operacji otrzymujemy dwa osobniki różniące się od swoich rodziców (rys. 7.26).



Rys. 7.25. Przykład kodowania drzewiastego



Rys. 7.26. Przykład krzyżowania w kodowaniu drzewiastym. Grubszą obwódką zaznaczono węzły wylosowane do krzyżowania

Mutacja jest trochę bardziej skomplikowaną operacją. Podobnie jak poprzednio, w chromosomie, na którym ma być przeprowadzona ta operacja, wybiera się losowo jeden węzeł. Na zawartości tego węzła można wykonać jedną z kilku operacji. Jeżeli jest to węzeł terminalny, można zmienić jego zawartość na inną, a węzeł ten dalej pozostaje węzłem terminalnym. Alternatywnie węzeł terminalny można zastąpić węzłem pośrednim z losowo wygenerowanym poddrzewem. Jeżeli wylosowany jest węzeł pośredni, można go zastąpić węzłem terminalnym. Wówczas usuwane jest całe poddrzewo węzła pośredniego. Inną możliwością jest zastąpienie węzła pośredniego innym węzłem z losowo wygenerowanym poddrzewem.

7.4. Zaawansowane techniki w algorytmach ewolucyjnych

7.4.1. Eksploracja i eksploatacja

W podrozdziale 7.3 poznaliśmy różne rodzaje algorytmów ewolucyjnych. Mimo różnic między nimi można zarysować wspólny schemat działania, który ogólnie sprowadza się do przetwarzania pewnej populacji osobników. Podczas selekcji, osobniki o lepszym przystosowaniu są powielane do nowej populacji z większym prawdopodobieństwem niż osobniki gorzej przystosowane. Dzięki tej operacji algorytm ewolucyjny ma tendencję podążania w kierunku lepszych rozwiązań. Ta zdolność poprawiania średniej wartości przystosowania populacji rodzicielskiej nazywa się *naciskiem selektywnym*. Mówimy, że algorytm charakteryzuje się dużym naciskiem selektywnym, gdy większa jest wartość oczekiwana liczby kopii lepszego osobnika niż wartość oczekiwana liczby kopii gorszego osobnika.

Nacisk selektywny jest ściśle powiązany z zależnością pomiędzy *eksploracją* a *eksploatacją* przestrzeni poszukiwań. Mówimy tu o dwóch skrajnych przypadkach, gdy z jednej strony algorytm ewolucyjny dokonuje eksploracji, czyli przeszukiwania całej przestrzeni rozwiązań w celu przybliżenia się do globalnego punktu będącego rozwiązaniem problemu. Drugim skrajnym przypadkiem jest eksploatacja, czyli poruszanie się po fragmencie przestrzeni w pobliżu globalnego rozwiązania. Eksplorację osiągamy poprzez zmniejszenie nacisku selektywnego. Wybierane są osobniki nie najlepiej przystosowane, ale mogące w przyszłości przybliżyć nas do optymalnego rozwiązania. Eksploatację natomiast uzyskujemy przez zwiększenie nacisku selektywnego, gdyż osobniki przechodzące do następnej populacji mają wartości coraz bliższe oczekiwanej, choć niekoniecznie globalnego rozwiązania. Wpływ na eksplorację i eksploatację mają także inne parametry algorytmu ewolucyjnego. Zwiększenie prawdopodobieństwa mutacji i krzyżowania powoduje większą różnorodność osobników, co, jak się domyślamy, prowadzi do poszerzenia obszaru poszukiwań i zapobiega przedwczesnej zbieżności. Zmniejszenie tych prawdopodobieństw w konsekwencji przyczynia się do tworzenia osobników coraz bardziej podobnych do siebie. Algorytm ewolucyjny może balansować między tymi dwoma skrajnymi przypadkami, a instrumentem do sterowania są właśnie jego parametry. Jeżeli zrównoważymy eksplorację i eksploatację, średnia wartość przystosowania osobników w populacji powinna się zwiększać, co może znacząco wpływać na skuteczność działania algorytmu ewolucyjnego.

7.4.2. Metody selekcji

Przedstawimy kilka metod selekcji najczęściej stosowanych w algorytmach genetycznych. Metody te polegają na wyborze osobników z populacji rodzicielskiej w celu dalszego przetwarzania za pomocą operatorów genetycznych. Jedna z nich, metoda koła ruletki, została już scharakteryzowana przy okazji omawiania klasycznego algorytmu genetycznego (punkt 7.3.1). Pozostałe metody, które zostaną przedstawione, to metoda rankingowa i turniejowa. Oczywiście oprócz nich istnieją inne metody selekcji.

7.4.2.1. Metoda koła ruletki

Metoda selekcji wykorzystująca mechanizm koła ruletki, mimo że jest procedurą losową, to jednak umożliwia wybór osobników rodzicielskich proporcjonalnie do wartości ich funkcji przystosowania, tzn. zgodnie z prawdopodobieństwem selekcji danym wzorem (7.2). Każdy z osobników otrzymuje w puli rodzicielskiej taką liczbę swoich kopii, jaką wynika ze wzoru

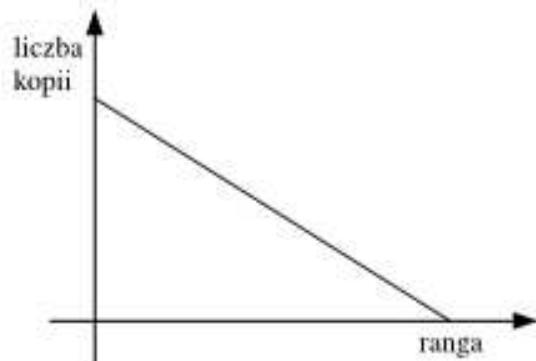
$$e(\text{ch}_i) = p_s(\text{ch}_i) \cdot K, \quad (7.30)$$

w którym K jest liczbą chromosomów $\text{ch}_i, i = 1, 2, \dots, K$ w populacji, a $p_s(\text{ch}_i)$ oznacza prawdopodobieństwo selekcji chromosomu ch_i dane wzorem (7.2). Ścisłe mówiąc, liczba kopii danego osobnika w puli rodzicielskiej równa się tyle, ile wynosi część całkowita $e(\text{ch}_i)$. Zauważmy, korzystając ze wzorów (7.2) i (7.30), że $e(\text{ch}_i) = F(\text{ch}_i)/\bar{F}$, gdzie \bar{F} jest średnią wartością funkcji przystosowania w populacji. Jest oczywiste, że z metody ruletki można korzystać, jeżeli wartości funkcji przystosowania są dodatnie. Metoda ta może mieć zastosowanie tylko w zadaniach maksymalizacji funkcji (a nie minimalizacji).

Problem minimalizacji można oczywiście łatwo sprowadzić do zagadnienia maksymalizacji funkcji i odwrotnie. Jednakże możliwość stosowania metody ruletki tylko do jednej klasy zadań, tzn. tylko maksymalizacji (lub tylko minimalizacji), jest niewątpliwie jej wadą. Drugą słabą stroną metody ruletki jest fakt, że osobniki o bardzo małej wartości funkcji przystosowania są zbyt wcześnie eliminowane z populacji, co może doprowadzić do przedwczesnej zbieżności algorytmu genetycznego. Aby temu zapobiec, stosuje się skalowanie funkcji przystosowania (punkt 7.4.3).

7.4.2.2. Selekcja rankingowa

W selekcji rankingowej (ang. *ranking selection*), zwanej też selekcją rangową, osobniki populacji są ustawiane kolejno w zależności od wartości ich funkcji przystosowania. Można to sobie wyobrazić jako listę rankingową osobników uszeregowanych od najlepiej do najgorzej przystosowanego (lub odwrotnie), gdzie każdemu osobnikowi przypisana jest liczba określająca jego kolejność na liście i nazywana *rangą* (ang. *rank*). Liczba kopii każdego osobnika, wprowadzonych do populacji rodzicielskiej $\mathbf{M}(t)$, jest ustalana zgodnie z wcześniej zdefiniowaną funkcją, zależną od rangi osobnika. Przykładową taką funkcję pokazuje rysunek 7.27.

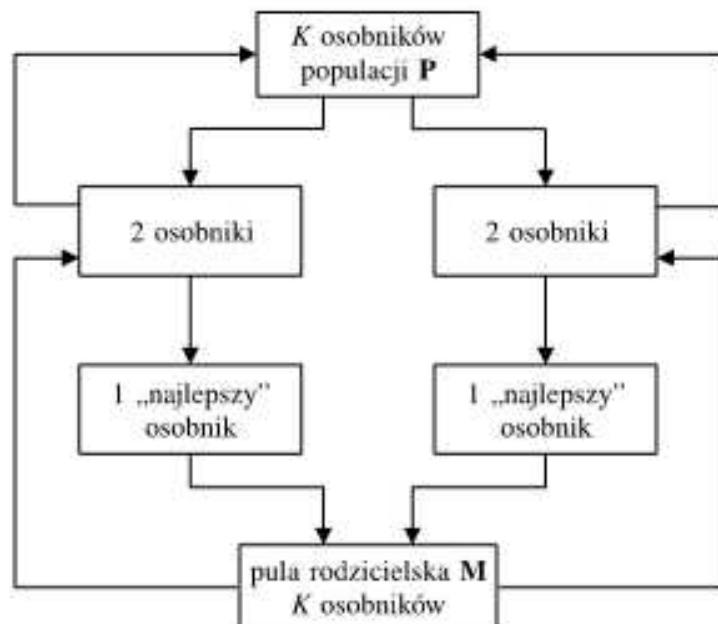


Rys. 7.27. Przykład funkcji określającej zależność liczby kopii osobnika w puli rodzicielskiej od jego rangi w selekcji rankingowej

Zaletą metody rankingowej jest możliwość wykorzystania jej zarówno do maksymalizacji, jak i do minimalizacji funkcji. Nie napotyka ona również konieczności skalowania w związku z problemem przedwczesnej zbieżności, co może wystąpić przy stosowaniu metody ruletki.

7.4.2.3. Selekcja turniejowa

W *selekcji turniejowej* (ang. *tournament selection*) dzieli się osobniki populacji na podgrupy, a następnie z każdej z nich wybiera się osobnika o najlepszym przystosowaniu. Rozróżnia się dwa sposoby takiego wyboru: *wybór deterministyczny* (ang. *deterministic tournament selection*) oraz *wybór losowy* (ang. *stochastic tournament selection*). W przypadku deterministycznym wyboru dokonuje się z prawdopodobieństwem równym 1, a w przypadku wyboru losowego — z prawdopodobieństwem mniejszym od 1. Podgrupy mogą być dowolnego rozmiaru, najczęściej dzieli się populację na podgrupy złożone z 2 lub 3 osobników. Metoda turniejowa nadaje się zarówno do problemów maksymalizacji, jak i minimalizacji funkcji. Oprócz tego może być łatwo rozszerzona na zadania dotyczące optymalizacji wielokryterialnej, a więc optymalizacji kilku funkcji jednocześnie. W metodzie turniejowej można zmieniać rozmiar podgrup, na jakie dzielona jest populacja (ang. *tournament size*). Badania wykazują, że metoda turniejowa działa lepiej niż metoda ruletki.



Rys. 7.28. Schematyczne przedstawienie selekcji turniejowej dla rozmiaru podgrup równego 2

Rysunek 7.28 przedstawia schemat ilustrujący metodę selekcji turniejowej dla podgrup złożonych z 2 osobników. Łatwo go uogólnić na większy rozmiar podgrup.

7.4.2.4. Inne metody selekcji

Istnieje wiele różnych odmian algorytmów selekcji. Przedstawione wcześniej metody (ruletki, turniejowa i rankingowa) są najczęściej stosowane. Inne stanowią pewną ich modyfikację lub połączenie.

Selekcja progowa jest szczególnym przypadkiem selekcji rankingowej, w której funkcja określająca prawdopodobieństwo przejścia osobnika do puli rodzicielskiej ma postać progu. Umożliwia to ustalenie odpowiedniego nacisku selektywnego, poprzez sterowanie wartością progu, od którego zależy wybór przystosowanych osobników.

Ciekawą metodą selekcji jest *selekcja stłoczenia* (ang. *crowding selection*), gdzie nowo utworzone osobniki zastępują najbardziej podobne do nich osobniki rodzicielskie, niezależnie od wartości ich funkcji przystosowania. Celem takiego postępowania jest zachowanie jak największej różnorodności populacji. Wprowadzony parametr *CF* (ang. *crowding factor*), określa liczbę rodziców podobnych do nowo powstałego osobnika, spośród których wylosowany będzie osobnik do usunięcia. Podobieństwo między nowymi i starymi osobnikami wyznacza miara Hamminga.

7.4.3. Skalowanie funkcji przystosowania

Skalowania funkcji przystosowania dokonuje się przede wszystkim z dwóch powodów. Po pierwsze, aby zapobiec przedwczesnej zbieżności algorytmu genetycznego. Po drugie, w końcowej fazie algorytmu, w przypadku gdy populacja zachowuje znaczną różnorodność, ale średnia wartość przystosowania niewiele odbiega od maksymalnej. Skalowanie funkcji przystosowania może wówczas zapobiec takiej sytuacji, że osobniki przeciętne i najlepsze otrzymują prawie taką samą liczbę potomków w następnych generacjach, co jest zjawiskiem niepożądany. Natomiast przedwczesna zbieżność algorytmu polega na tym, że najlepsze, ale jeszcze nie optymalne chromosomy dominują w populacji. Zjawisko to może wystąpić w algorytmie z selekcją metodą koła ruletki. W ciągu kilku generacji przy selekcji proporcjonalnej do wartości funkcji przystosowania populacja będzie zawierała tylko kopie najlepszego chromosomu z populacji początkowej. Jest nieprawdopodobne, aby taki chromosom reprezentował optymalne rozwiązanie, skoro populacja początkowa jest tylko małą próbą losową z całej przestrzeni poszukiwań. Skalowanie funkcji przystosowania chroni populację przed dominacją nieoptymalnego chromosomu, a zatem zapobiega przedwczesnej zbieżności algorytmu ewolucyjnego.

Skalowanie polega na odpowiednim przekształceniu funkcji przystosowania. Rozróżniamy 3 zasadnicze metody skalowania: skalowanie liniowe, obcinanie typu sigma i skalowanie potęgą.

7.4.3.1. Skalowanie liniowe

Skalowanie liniowe (ang. *linear scaling*) polega na przekształceniu funkcji przystosowania F do postaci F' poprzez następującą zależność liniową:

$$F' = a \cdot F + b, \quad (7.31)$$

w której a i b są stałymi dobieranymi w taki sposób, aby średnia wartość funkcji przystosowania po skalowaniu była równa średniej wartości funkcji przystosowania przed skalowaniem, a maksymalna wartość funkcji przystosowania po skalowaniu była wielokrotnością średniej wartości funkcji przystosowania. Współczynnik wielokrotności przyjmuje się często między 1,2 a 2. Należy zwrócić uwagę, aby funkcja F' nie przyjmowała wartości ujemnych.

7.4.3.2. Obcinanie typu sigma

Obcinanie typu sigma (ang. *sigma truncation*) jest metodą skalowania polegającą na przekształceniu funkcji przystosowania F do postaci F' według następującej zależności:

$$F' = F + (\bar{F} - c \cdot \sigma), \quad (7.32)$$

w której \bar{F} jest średnią wartością funkcji przystosowania w populacji, c jest małą liczbą naturalną (zwykle od 1 do 5), a σ jest odchyleniem standardowym w populacji. Jeśli wystąpią wartości ujemne F' , to przyjmuje się je jako równe zeru.

7.4.3.3. Skalowanie potęgą

Skalowanie potęgą (ang. *power law scaling*) jest metodą skalowania przekształcającą funkcję przystosowania F zgodnie z zależnością:

$$F' = F^k, \quad (7.33)$$

w której k jest liczbą bliską 1. Wybór k zwykle zależy od problemu. Można przyjąć np. $k = 1,005$.

W pracy [75] opisano także inne rodzaje skalowania, jak np. skalowanie logarytmiczne, „metodą okna”, metodą Boltzmanna oraz rankingowe liniowe i rankingowe wykładnicze.

7.4.4. Szczególne procedury reprodukcji

Jako szczególne procedury reprodukcji można potraktować strategię elitarną oraz algorytm genetyczny z częściową wymianą populacji.

7.4.4.1. Strategia elitarna

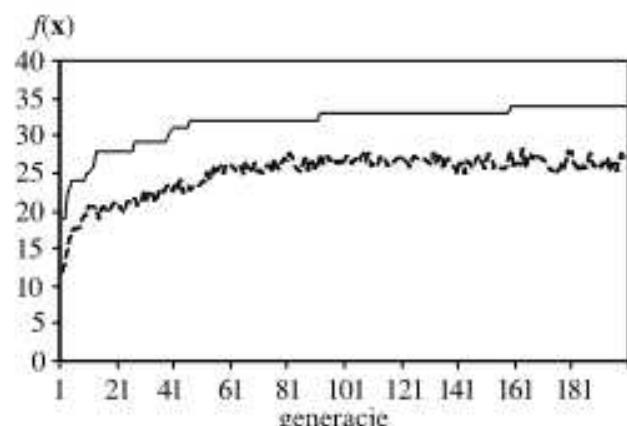
Strategia elitarna (ang. *elitist strategy*) polega na ochronie najlepszych chromosomów w kolejnych generacjach. W klasycznym algorytmie genetycznym najlepiej przystosowane osobniki nie zawsze przechodzą do następnej generacji. Nie zawsze nowa populacja $\mathbf{P}(t+1)$ zawiera chromosom o największej wartości funkcji przystosowania z populacji $\mathbf{P}(t)$. Strategia elitarna stosowana jest w celu zabezpieczenia przed utratą takiego osobnika. Jest on zawsze włączony do nowej populacji.

7.4.4.2. Algorytm genetyczny z częściową wymianą populacji

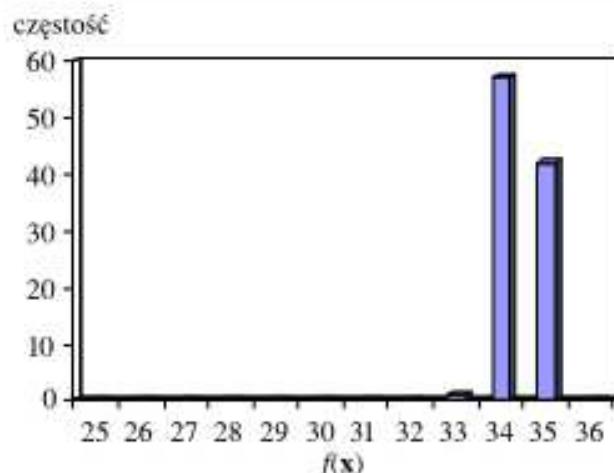
Algorytm genetyczny z częściową wymianą populacji, nazywany też *algorytmem z ustaloną stanem* (ang. *steady-state*), charakteryzuje się tym, że część populacji przechodzi do następnej generacji bez jakichkolwiek zmian. Oznacza to, że ta część populacji nie podlega operacjom krzyżowania i mutacji. Często w konkretnej implementacji tego typu algorytmu tylko jeden lub dwa osobniki są wymieniane w danej chwili zamiast krzyżowania i mutacji w obrębie całej populacji.

Przykład 7.16

Ponownie rozwiążemy problem plecakowy opisany w przykładzie 7.6. Tym razem zastosujemy metodę koła ruletki oraz strategię elitarną. Strategia elitarna będzie polegać



Rys. 7.29. Wykres maksymalnej (linia ciągła) oraz średniej w populacji (linia przerywana) wartości funkcji przystosowania w 200 kolejnych generacjach



Rys. 7.30. Histogram rozwiązań uzyskiwanych w przeprowadzonych 100 próbach algorytmu genetycznego

na automatycznym przechodzeniu do następnej populacji 2 osobników najlepiej przystosowanych. Na rysunku 7.29 przedstawiono wykresy wartości funkcji przystosowania najlepszego osobnika (linia ciągła) oraz średniej wartości wszystkich osobników populacji (linia przerywana) w kolejnych generacjach algorytmu genetycznego. Łatwo zauważać, że zmiany wartości funkcji przystosowania najlepszego osobnika nie zachodzą już tak chaotycznie, jak w przykładzie 7.6. W kolejnych generacjach najlepsze rozwiązania nie są tracone i dlatego na rysunku 7.29 maksymalna wartość funkcji przystosowania rośnie lub się nie zmienia. Warto zwrócić uwagę na jeszcze jedną korzyść z zastosowania strategii elitarnej. Znalezione zostały nowe rozwiązania, które pozwalają zapakować do plecaka 34 lub 35 przedmiotów. Prześledźmy teraz histogram (rys. 7.30) rozwiązań w 100 kolejnych uruchomieniach algorytmu genetycznego i porównajmy z podobnym wykresem (rys. 7.6) z przykładu 7.6. Widać wyraźną poprawę rezultatów. Znaleziono tylko jedno rozwiązanie z 33 przedmiotami (poprzednio było ono rozwiązaniem najlepszym), 57 rozwiązań wskazywało na 34 przedmioty i w 42 przypadkach chromosom był rozwiązaniem pozwalającym na zapakowanie do plecaka 35 przedmiotów.

7.4.5. Metody kodowania

W klasycznym algorytmie genetycznym stosuje się binarne kodowanie chromosomów. Wykorzystuje się znany sposób zapisu liczb dziesiętnych w systemie dwójkowym, gdzie każdy bit kodu dwójkowego odpowiada kolejnej potędze liczby 2. Przykładowo ciąg binarny [10011] jest kodem liczby 19, gdyż $1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$. W przypadku kodowania liczb rzeczywistych wartość zmiennej niezależnej $x_i \in [a_i, b_i] \in R$, zakodowanej za pomocą n_i bitów, wyznacza się na podstawie wartości genów chromosomu \mathbf{x} , o kolejnych numerach począwszy od $s(i)$, a skończywszy na $s(i) + n_i - 1$

$$x_i = a_i + \frac{b_i - a_i}{2^{n_i} - 1} \sum_{j=0}^{n_i-1} 2^j x_{s(i)+j}. \quad (7.34)$$

Powyzszy wzór wynika z prostego odwzorowania liniowego przedziału $[a_i, b_i]$ na przedział $[0, 2^{n_i} - 1]$, gdzie $2^{n_i} - 1$ jest liczbą dziesiętną, zakodowaną w postaci ciągu binarnego o długości n_i złożonego z samych jedynek, a 0 jest oczywiście wartością dziesiętną ciągu binarnego o długości n_i złożonego z samych zer.

W algorytmach genetycznych można stosować na przykład kod Graya, charakteryzujący się tym, że ciągi binarne odpowiadające dwóm kolejnym liczbom całkowitym różnią się tylko jednym bitem. Taki sposób kodowania chromosomów może okazać się uzasadniony ze względu na operację mutacji.

Kodowanie logarytmiczne (ang. *logarithmic coding*) stosowane jest w celu zmniejszenia długości chromosomów w algorytmie genetycznym. Wykorzystuje się je głównie w problemach optymalizacyjnych z wieloma parametrami, o dużych przestrzeniach poszukiwań.

W kodowaniu logarytmicznym pierwszy bit (α) ciągu kodowego jest bitem znaku funkcji wykładniczej, drugi bit (β) jest bitem znaku wykładnika funkcji wykładniczej, a pozostałe bity (bin) są reprezentacją wykładnika funkcji wykładniczej

$$[\alpha \beta bin] = (-1)^\beta e^{(-1)^\alpha [bin]_{10}}, \quad (7.35)$$

przy czym $[bin]_{10}$ oznacza wartość dziesiętną liczby zakodowanej w postaci ciągu binarnego bin .

Przykład 7.17

Łatwo sprawdzić, że [10110] jest ciągiem kodowym liczby

$$x_1 = (-1)^0 e^{(-1)^1 [110]_{10}} = e^{-6} = 0,002478752.$$

Podobnie [01010] jest ciągiem kodowym liczby

$$x_2 = (-1)^1 e^{(-1)^0 [010]_{10}} = -e^2 = -7,389056099.$$

Zauważmy, że w ten sposób za pomocą 5 bitów można zakodować liczby z przedziału $[-e^7, e^7]$. Jest to znacznie większy zakres niż $[0, 31]$ stosowany w kodowaniu binarnym.

Zaletą kodowania binarnego jest prostota w stosowaniu i możliwość użycia prostych operatorów genetycznych, ale są również i wady. Alfabet binarny nie jest naturalny dla większości problemów optymalizacyjnych. Powoduje też powstawanie olbrzymich przestrzeni rozwiązań. Nawet zakodowanie kilku zmiennych niezależnych w chromosomie, przy dość dużej dokładności (znaczna liczba bitów przeznaczona do zakodowania wartości zmiennych), powoduje gwałtowny rozrost długości chromosomu. O wiele lepsze wydaje się kodowanie liczbami zmiennoprzecinkowymi, szczególnie w problemach optymalizacji numerycznej.

Przykład 7.18

Zastosujemy klasyczny algorytm genetyczny do znalezienia maksimum funkcji

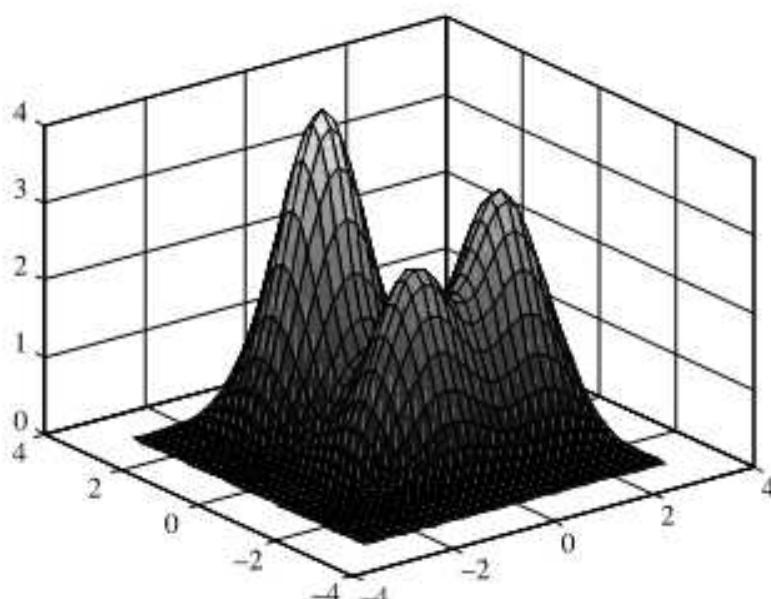
$$f(x_1, x_2) = 2(1 - x_1)e^{-x_1^2 - x_2^2} + 3e^{-x_1^2 - (x_2 - 2)^2} + 4e^{-(x_1 - 2)^2 - x_2^2} \quad (7.36)$$

przy ograniczeniach $-3 < x_1 < 3$ i $-3 < x_2 < 3$. Wykres tej funkcji przedstawia rysunek 7.31. Zmienne x_1 i x_2 kodujemy za pomocą odwzorowania liniowego (7.34), wykorzystując 20 bitów na każdą zmienną. Na przykład dla chromosomu

[00010111101101011101 00011001100101101000]

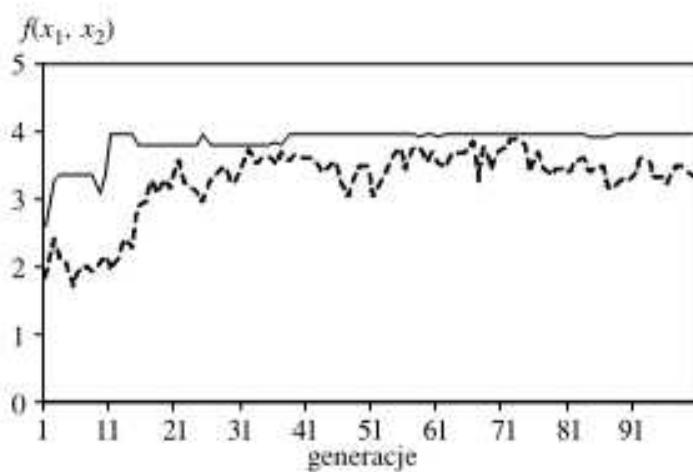
możemy wyznaczyć fenotyp

[1,3797, -2,4703],



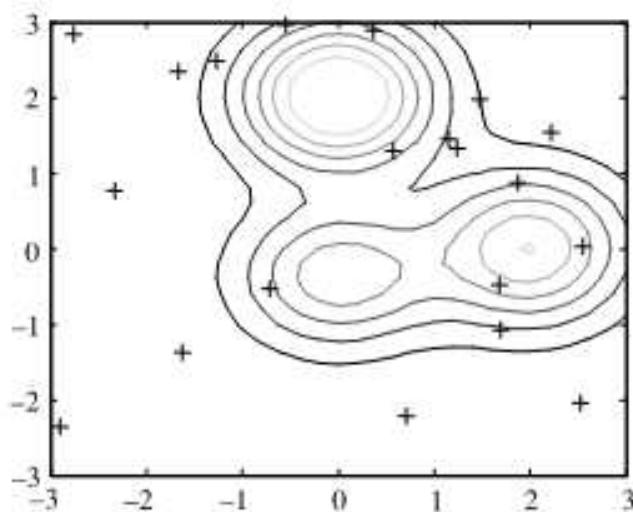
Rys. 7.31. Wykres funkcji przedstawionej w przykładzie 7.18

przy czym $a_i = -3$, $b_i = 3$, $i = 1, 2$. Przyjmujemy prawdopodobieństwo mutacji $p_m = 0.01$ i prawdopodobieństwo krzyżowania $p_k = 0.7$, natomiast populacja liczy 20 osobników. Algorytm zostanie zatrzymany po 100 generacjach. Zmiany maksymalnej (linia ciągła) oraz średniej (linia przerywana) wartości funkcji przystosowania można zaobserwować na rysunku 7.32.

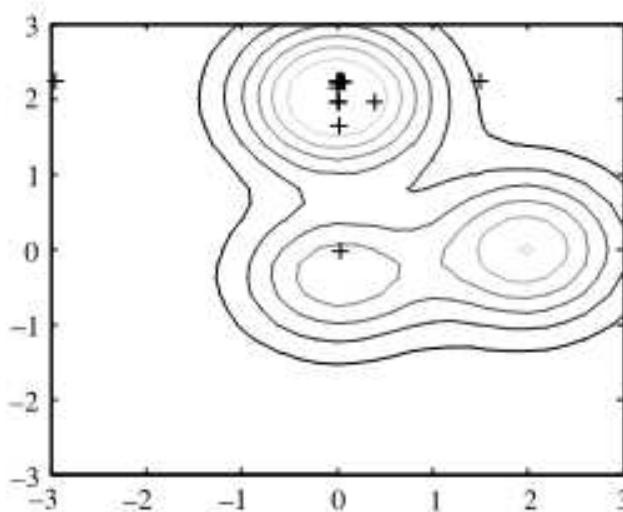


Rys. 7.32. Wykres maksymalnej (linia ciągła) oraz średniej w populacji (linia przerywana) wartości funkcji przystosowania w 100 kolejnych generacjach

Widać, że rozwiązanie zostało znalezione dość szybko, bo już w 37. generacji. Przyjrzymy się wykresom konturowym na rysunkach 7.33 i 7.34. Na rysunku 7.33 pokazano początkowo wygenerowane rozwiązania. Krzyżykami zaznaczono punkty, które są rozwiązaniami zakodowanymi w poszczególnych chromosomach. Zauważmy, że pod koniec pracy algorytmu (rys. 7.34) większość osobników znajduje się w pobliżu poszukiwanego maksimum.



Rys. 7.33. Rozkład rozwiązań po wygenerowaniu początkowej populacji



Rys. 7.34. Rozkład rozwiązań po 100 generacjach

7.4.6. Rodzaje krzyżowań

W klasycznym algorytmie genetycznym operacją krzyżowania jest tzw. krzyżowanie jednopunktowe, przedstawione w punkcie 7.3.1. Stosuje się również inne rodzaje krzyżowania: dwupunktowe, wielopunktowe bądź równomierne.

7.4.6.1. Krzyżowanie dwupunktowe

Krzyżowanie dwupunktowe (ang. *two-point crossover*), jak nazwa wskazuje, tym różni się od krzyżowania jednopunktowego, że potomkowie dziedziczą fragmenty chromosomów rodzicielskich wyznaczone przez 2 wylosowane punkty krzyżowania.

Przykład 7.19

Rozważmy dwa chromosomy $ch_1 = [1101001110]$ i $ch_2 = [1011111100]$, które poddamy krzyżowaniu dwupunktowemu. Wylosowano dwa punkty krzyżowania: 2 i 7. Oto jak wygląda proces krzyżowania.

Para rodziców:

$$\begin{array}{c} ch_1 = [11 \mid 01001 \mid 110] \\ ch_2 = [10 \mid 11111 \mid 100] \end{array} \xrightarrow{\text{krzyżowanie}} \quad$$

Para potomków:

$$\begin{array}{c} [11 \mid \mathbf{11111} \mid 110] \\ [10 \mid \mathbf{01001} \mid 100] \end{array}$$

Syntezą | oznaczono miejsce krzyżowania, a pogrubioną czcionką zaznaczono zamienione fragmenty chromosomów.

7.4.6.2. Krzyżowanie wielopunktowe

Krzyżowanie wielopunktowe (ang. *multiple-point crossover*) jest uogólnieniem poprzednich operacji i charakteryzuje się odpowiednio większą liczbą punktów krzyżowania.

Przykład 7.20

Dokonamy krzyżowania wielopunktowego dla czterech punktów krzyżowania, na chromosomach ch_1 i ch_2 , które przedstawiono w przykładzie 7.19. Wylosowano następujące

miejsca krzyżowania: 1, 4, 6, 9. Krzyżowanie przebiega następująco:

Para rodziców:		Para potomków:
$ch_1 = [1 101 00 111 0]$	krzyżowanie	$[1 \mathbf{011} 00 \mathbf{110} 0]$
$ch_2 = [1 011 11 110 0]$	\longrightarrow	$[1 \mathbf{101} 11 \mathbf{111} 0]$

Przykład 7.21

Dla chromosomów ch_1 i ch_2 , przedstawionych w przykładzie 7.19, dokonamy operacji krzyżowania wielopunktowego, przy uwzględnieniu trzech punktów krzyżowania. Wylosowano następujące punkty krzyżowania 4, 6 i 8. Krzyżowanie przebiega następująco:

Para rodziców:		Para potomków:
$ch_1 = [1101 00 11 10]$	krzyżowanie	$[1101 \mathbf{11} 11 \mathbf{00}]$
$ch_2 = [1011 11 11 00]$	\longrightarrow	$[1011 \mathbf{00} 11 \mathbf{10}]$

Analogicznie odbywa się krzyżowanie dla 5 oraz większej nieparzystej liczby punktów krzyżowania. Krzyżowanie jednopunktowe jest oczywiście szczególnym przypadkiem takiego krzyżowania.

7.4.6.3. Krzyżowanie równomierne

Krzyżowanie równomierne (ang. *uniform crossover*), nazywane też *krzyżowaniem jednolitym* lub *jednostajnym*, odbywa się zgodnie z wylosowanym wzorcem wskazującym, które geny dziedziczone są od pierwszego z rodziców (pozostałe pochodzą od drugiego). Krzyżowanie to można stosować do różnych rodzajów kodowania chromosomu. Musi być spełniony tylko jeden warunek — chromosomy muszą być tej samej długości.

Przykład 7.22

Założymy, że dla tej samej pary rodziców co w przykładzie 7.19 wylosowano następujący wzorzec: 0101101110, w którym 1 oznacza przejęcie genu na odpowiedniej pozycji (locus) od rodzica 1, a 0 — od rodzica 2. Wtedy krzyżowanie równomierne przebiega w następujący sposób:

Para rodziców:		Para potomków:
$ch_1 = [1101001110]$	krzyżowanie	$[1\mathbf{001101100}]$
$ch_2 = [1011111100]$	\longrightarrow	$[111\mathbf{1011110}]$

Wylosowany wzorzec:

0101101110

Pogrubioną czcionką zaznaczono zamienione geny.

7.4.7. Rodzaje mutacji

Klasyczny algorytm genetyczny jest wyposażony w operator mutacji. Ma on za zadanie wprowadzenie pewnego urozmaicenia w populacji, jednak jego rola jest raczej mała.

W innych rodzajach algorytmów ewolucyjnych jest on operatorem dominującym. Opiszemy teraz kilka sposobów mutacji.

7.4.7.1. Mutacja w przypadku kodowania binarnego

Jeden z rodzajów mutacji przeznaczonej do kodowania binarnego poznaliśmy w punkcie 7.3.1, przy okazji omawiania klasycznego algorytmu genetycznego. Operacjami mutacji podlegały te geny w chromosomie, dla których wylosowana liczba z zakresu $[0, 1]$ była mniejsza od prawdopodobieństwa mutacji p_m . Mutacja może polegać na negacji wartości bitu lub na zamianie bitu na wylosowaną wartość ze zbioru $\{0, 1\}$.

7.4.7.2. Mutacja w przypadku kodowania liczbami zmiennoprzecinkowymi

Jeżeli chromosom jest kodowany za pomocą liczb rzeczywistych, nie można przeprowadzić prostej negacji. Stosuje się pewne uogólnienie mutacji binarnej. Zakładamy, że wartość i -tego genu x_i jest ograniczona w przedziale $[a_i, b_i]$. Dla każdego genu losujemy liczbę z zakresu $[0, 1]$ i jeżeli jest ona mniejsza od prawdopodobieństwa mutacji p_m , to przeprowadzamy mutację według wzoru

$$y_i = a_i + (b_i - a_i)U_i(0, 1), \quad (7.37)$$

w którym y_i to nowa wartość genu, a $U_i(0, 1)$ — zmienna losowa wygenerowana z rozkładu równomiernego w przedziale $(0, 1)$. Zauważmy, że wartość y_i nie zależy od x_i , więc im większa wartość prawdopodobieństwa p_m , tym bardziej działanie tego operatora upodabnia się do losowego próbkowania przestrzeni poszukiwań.

Częściej stosowana jest mutacja polegająca na dodaniu do każdej wartości genu x_i z_i pewnej zmiennej losowej Z_i , tzn.

$$y_i = x_i + z_i. \quad (7.38)$$

Najczęściej stosowane rozkłady to rozkład normalny lub rozkład Cauchy'ego.

7.4.8. Inwersja

Holland [82] przedstawia trzy techniki pozwalające na otrzymanie potomków różniących się od chromosomów rodzicielskich. Są to: krzyżowanie, mutacja i inwersja. Inwersja działa na pojedynczym chromosomie, zmieniając kolejność alleli między dwoma losowo wybranymi pozycjami (locus) chromosomu. Mimo że operator ten został zdefiniowany przez analogię do biologicznego procesu inwersji chromosomalnej, to jednak nie jest zbyt często stosowany w algorytmach genetycznych.

Przykład 7.23

Jako przykład działania inwersji rozważmy chromosom $[001100111010]$. Założymy, że wylosowano następujące pozycje: 4 oraz 10. Po dokonaniu inwersji otrzymamy chromosom $[00110\mathbf{11100}10]$, gdzie pogrubioną czcionką zaznaczono zmienione geny.

7.5. Algorytmy ewolucyjne w projektowaniu sieci neuronowych

Wykorzystanie sieci neuronowych do rozwiązania jakiegokolwiek zadania wymaga wcześniejszego ich zaprojektowania. Należy dobrać odpowiednią architekturę sieci, ustalając liczbę warstw, liczbę neuronów w każdej z nich i sposób ich połączenia. Następnie przeprowadzany jest proces uczenia, z wykorzystaniem na przykład algorytmu wstępnej propagacji błędów. Czynności te wymagają czasami sporo czasu i doświadczenia, ale z pomocą mogą nam przyjść algorytmy ewolucyjne.

Metody ewolucyjne możemy zastosować do rozwiązywania następujących zadań:

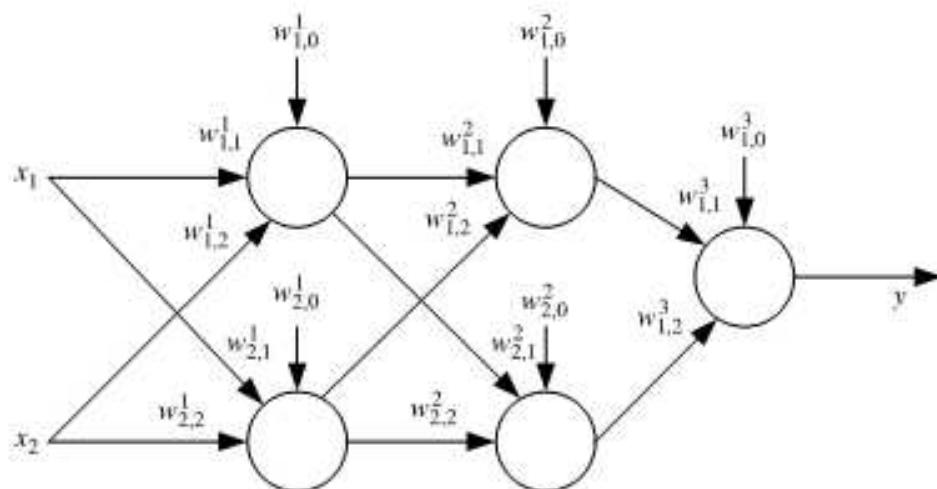
- do uczenia wag sieci neuronowych,
- do poszukiwania optymalnej architektury,
- do równoczesnego określania struktury i wartości wag.

Pomysł, że sieci neuronowe mogą być uczone za pomocą algorytmu ewolucyjnego, pojawił się u wielu badaczy. Pierwsze prace na ten temat dotyczyły zastosowania algorytmu genetycznego jako metody uczenia małych jednokierunkowych sieci neuronowych, następnie odnotowano pomyślne wykorzystanie tego algorytmu w przypadku względnie dużych sieci. Dwa najważniejsze argumenty przemawiające za stosowaniem algorytmów ewolucyjnych do problemów optymalizacji wag sieci neuronowej to przede wszystkim globalne przeszukiwanie przestrzeni wag i unikanie minimów lokalnych. Poza tym, algorytmy ewolucyjne mogą być wykorzystywane w problemach, w których uzyskanie informacji dotyczącej gradientów jest trudne lub kosztowane. Ewolucyjnemu uczeniu wag sieci neuronowych poświęcony jest podrozdział 7.5.1. Optymalne projektowanie architektury sieci neuronowej można traktować jako poszukiwanie struktury, która działa najlepiej dla określonego zadania. Oznacza to przeszukiwanie przestrzeni architektur i wybór najlepszego spośród elementów tej przestrzeni, korzystając z określonego kryterium optymalności. Typowy cykl ewolucji architektur jest przedstawiony w punkcie 7.5.2.

Metody jednoczesnego ewolucyjnego uczenia wag sieci neuronowej i poszukiwania optymalnej architektury możemy połączyć w ramach jednego algorytmu ewolucyjnego. Zarys tej problematyki przedstawiony zostanie w punkcie 7.5.3.

7.5.1. Algorytmy ewolucyjne do uczenia wag sieci neuronowych

Uczenie sieci neuronowej polega na wyznaczeniu wartości wag sieci, której topologia została wcześniej ustalona. Wagi są kodowane w chromosomie w postaci ciągu binarnego bądź wektora liczb rzeczywistych. Każdy osobnik populacji jest określony przez całkowity zbiór wag sieci neuronowej. Przykład takiego chromosomu przedstawia rysunek 7.35. W chromosomie tym odwzorowano wagę sieci z dwoma wejściami, dwiema ukrytymi warstwami, dwoma neuronami w każdej z tych warstw i jednym neuronem w warstwie wyjściowej. Łącznie z wagami polaryzacji chromosom zawiera informacje o 15 wagach. Kolejność umieszczenia wag w chromosomie jest dowolna, ale nie może być zmieniona z chwilą rozpoczęcia uczenia.



Chromosom z zakodowanymi wagami:

$w_{1,0}^1$	$w_{1,1}^1$	$w_{1,2}^1$...	$w_{2,2}^2$	$w_{1,0}^3$	$w_{1,1}^3$	$w_{1,2}^3$
-------------	-------------	-------------	-----	-------------	-------------	-------------	-------------

Rys. 7.35. Odwzorowanie wag sieci neuronowej w chromosomie dla przykładowej sieci

Ocena przystosowania osobników dokonywana jest na podstawie funkcji przystosowania zdefiniowanej jako suma kwadratów błędów, będących różnicami między sygnałem wzorcowym a sygnałem wyjściowym sieci dla różnych danych wejściowych. Zajmijmy się teraz wyborem schematu reprezentacji wag. Należy się zdecydować na reprezentację binarną lub na przedstawienie wag w postaci liczb rzeczywistych. Do binarnego kodowania wag można zastosować, oprócz naturalnego kodu dwójkowego, także kod Graya, kodowanie logarytmiczne (punkt 7.4.5) lub inne bardziej złożone metody kodowania. Ograniczeniem jest dokładność reprezentacji wag. Jeżeli zbyt mało bitów użyto do przedstawienia każdej wagi, to uczenie może trwać zbyt długo, a nawet nie przynieść efektu, ponieważ niektóre kombinacje rzeczywistych wag nie będą mogły być aproksymowane przez dyskretne wartości w pewnym zakresie tolerancji. Z drugiej strony, jeżeli użyjemy zbyt wielu bitów, to ciągi binarne reprezentujące duże sieci neuronowe będą bardzo długie, co znacznie przedłuży proces ewolucji i uczyni ewolucyjne podejście do uczenia niepraktycznym. Aby uniknąć wad schematu reprezentacji binarnej, zaproponowano reprezentację wag za pomocą liczb rzeczywistych, tzn. jedna liczba rzeczywista do przedstawienia jednej wagi.

Po wybraniu schematu chromosomalnego reprezentacji, na przykład tak, jak pokazano na rysunku 7.35, algorytm ewolucyjny działa na populacji osobników (chromosomów reprezentujących sieci neuronowe o tej samej architekturze, ale różnych wartościach wag) według typowego cyklu ewolucji obejmującego następujące kroki:

- 1) Dekodowanie każdego osobnika bieżącej populacji do zbioru wag i skonstruowanie odpowiadającej mu sieci neuronowej z tym zbiorem wag, przy czym architektura sieci i reguła uczenia są wcześniej ustalone.
- 2) Obliczanie całkowitego błędu średniego kwadratowego różnicy między sygnałami wzorcowymi i wyjściowymi dla wszystkich sieci. Błąd ten określa przystosowanie osobnika (skonstruowanej sieci); można też inaczej zdefiniować funkcję przystosowania w zależności od rodzaju sieci.

3) Reprodukcja osobników z prawdopodobieństwem odpowiednim do ich przystosowania lub rangi, w zależności od użytej metody selekcji.

4) Zastosowanie operatorów genetycznych, takich jak krzyżowanie, mutacja i/lub inwersja i otrzymanie nowej generacji.

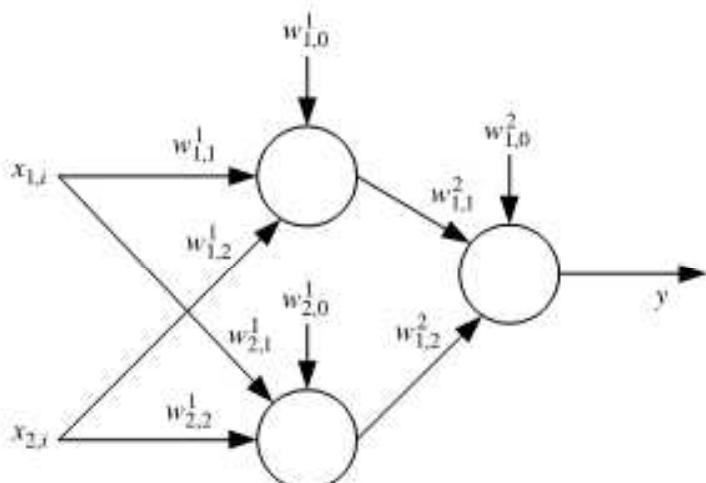
Przykład 7.24

Rozważmy sieć neuronową przedstawioną na rysunku 7.36. Sieć tę wykorzystamy do realizacji układu XOR (patrz rozdział 6). Wyznaczmy wagi $w_{1,1}^1, w_{1,2}^1, w_{2,1}^1, w_{2,2}^1, w_{1,1}^2, w_{1,2}^2$ oraz $w_{1,0}^1, w_{2,0}^1, w_{1,0}^2$, które minimalizują błąd

$$Q = \frac{1}{4} \sum_{i=1}^4 (d_i - y_i)^2.$$

Sygnał wyjściowy dany jest wzorem

$$y_i = \left(\frac{1}{1 + \exp((-\beta)(w_{1,1}^2(1/(1 + \exp((-\beta)(w_{1,1}^1 x_{1,i} + w_{1,2}^1 x_{2,i} + w_{1,0}^1)))) + w_{1,2}^2(1/(1 + \exp((-\beta)(w_{2,1}^1 x_{1,i} + w_{2,2}^1 x_{2,i} + w_{2,0}^1)))) + w_{1,0}^2))} \right).$$



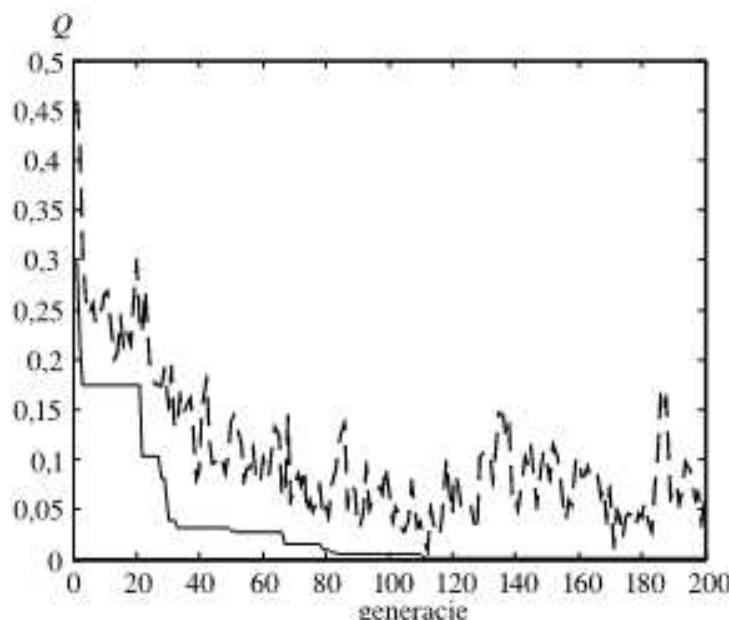
Rys. 7.36. Sieć realizująca układ XOR

Zadanie sprowadza się do znalezienia wartości dziesięciu wyżej wymienionych wag, dla których sieć nauczy się poprawnie wykonywać zadanie XOR. Informacje w chromosomie będą przechowywane analogicznie, jak pokazano na rysunku 7.35. Do wyznaczania optymalnych wag wykorzystamy klasyczny algorytm genetyczny, zaimplementowany w programie *FlexTool* [54].

Populacja składa się z 31 osobników. Jako metodę selekcji wybrana została metoda koła ruletki. Każda wartość wagi, jako liczba całkowita w zakresie $[-10, 10]$, jest kodowana binarnie za pomocą pięciu bitów. Prawdopodobieństwa krzyżowania i mutacji wynoszą, odpowiednio, $p_k = 0,77$ i $p_m = 0,0077$, przy czym krzyżowanie przebiega w dwóch punktach. Algorytm genetyczny działa przez 200 generacji.

Rezultatem działania algorytmu genetycznego jest znalezienie wag o wartościach: $w_{1,1}^1 = -10, w_{1,2}^1 = 9, w_{2,1}^1 = 8, w_{2,2}^1 = -9, w_{1,1}^2 = 10, w_{1,2}^2 = 10$ oraz $w_{1,0}^1 = -4, w_{2,0}^1 = -4, w_{1,0}^2 = -4$. Błąd średni kwadratowy dla tych wartości wag wynosi $Q = 2,7260 \cdot 10^{-4}$. Jak widać, sieć realizuje postawione przed nią zadanie z dość małą

wartością tego błędu. Rysunek 7.37 przedstawia wartości błędu średniego kwadratowego dla kolejnych generacji.



Rys. 7.37. Przebieg najmniejszej (linia ciągła) i średniej (linia przerywana) wartości błędu średniego kwadratowego do przykładu 7.24

7.5.2. Algorytmy ewolucyjne do określania topologii sieci neuronowej

W poprzednim punkcie, dotyczącym ewolucyjnego uczenia sieci neuronowej, zakładano, że architektura sieci jest wcześniej ustalona i nie zmienia się podczas procesu ewolucji wag. Istotne jest jednak pytanie, w jaki sposób dokonać wyboru architektury sieci. Wiadomo, że architektura ma decydujący wpływ na przetwarzanie informacji przez sieć neuronową. Niestety, najczęściej jest ona tworzona przez ekspertów metodą prób i błędów. Warto zastanowić się nad możliwością automatycznego sposobu projektowania architektury sieci neuronowej do rozwiązania określonego zadania. Takim sposobem może być ewolucyjne projektowanie architektury, wykorzystujące algorytm ewolucyjny.

Podobnie jak w przypadku ewolucyjnego uczenia, pierwszym etapem ewolucyjnego projektowania architektury jest podjęcie decyzji dotyczącej odpowiedniej jej reprezentacji w chromosomie. Jednak w tym przypadku problem nie dotyczy wyboru między reprezentacją binarną a rzeczywistą (liczby rzeczywiste), gdyż mamy do czynienia tylko z wartościami dyskretnymi. Obecnie zagadnienie to związane jest bardziej z koncepcją struktury reprezentacji, tj. macierz, graf czy pewne ogólne reguły. Ogólnie rodzaje kodowania można podzielić na *kodowanie bezpośrednie* i *kodowanie pośrednie*. Kodowanie bezpośrednie polega na reprezentowaniu w chromosomie najmniejszych jednostek, jakie wybierzymy, by określić budowę sieci neuronowej. Mogą to być połączenia, neurony (węzły sieci) lub warstwy. W zależności od wyboru możemy rozróżnić następujące kodowania sieci w chromosomach:

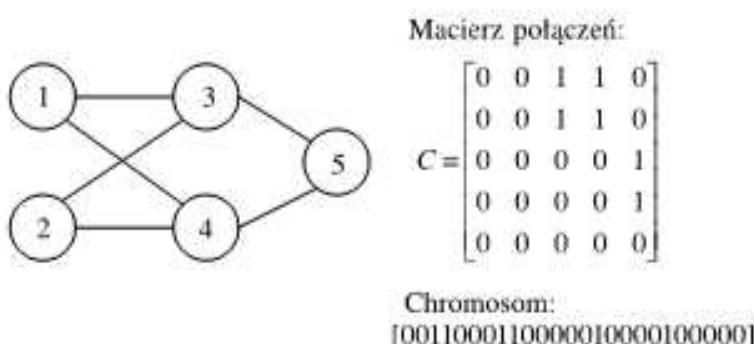
- **Kodowanie bazujące na połączeniach** — jest to jeden z pierwszych sposobów reprezentowania struktury sieci. Chromosom jest łańcuchem wartości wag lub informacją o występowaniu połączeń. Takie podejście wymaga ustalenia maksymalnego rozmiaru architektury, w której jest określona liczba połączeń oraz neuronów i warstw, mogących występować w największej sieci neuronowej.

• **Kodowanie bazujące na węzłach** — w metodzie tej chromosom reprezentuje łańcuch lub drzewo węzłów. Każda informacja o węźle zakodowana w chromosomie może zawierać jego względową pozycję, połączenia z poprzednimi warstwami, funkcję aktywacji i inne. Przeprowadzając operację krzyżowania i mutacji, należy zadbać, aby wymienić całą informację o zakodowanych neuronach.

• **Kodowanie oparte na warstwach** — w metodzie tej podstawową informacją kodowaną w chromosomie jest warstwa. Metodę tę można wykorzystać do projektowania większych sieci. Schemat kodowania jest skomplikowanym opisem połączeń między warstwami, dlatego wymagane są specjalne operatory genetyczne.

• **Kodowania bazujące na ścieżkach** — sieć jest przedstawiana jako zbiór ścieżek od wejścia do wyjścia neuronu. Ten sposób kodowania można wykorzystać przy projektowaniu rekurencyjnych sieci neuronowych. Tutaj również używane są specjalne operatory algorytmów genetycznych.

Najprostszą metodą bezpośredniego kodowania jest kodowanie połączeń w sieci neuronowej za pomocą macierzy połączeń. Macierz C o wymiarze $n \times n$, $C = [c_{ij}]_{n \times n}$ może reprezentować połączenia sieci neuronowej o n węzłach (neuronach), gdzie c_{ij} oznacza obecność lub brak połączenia między neuronami i oraz j , tzn. $c_{ij} = 1$, jeśli takie połączenie istnieje, a w przypadku braku takiego połączenia $c_{ij} = 0$. Ciąg binarny (chromosom) reprezentujący połączenia w sieci neuronowej jest po prostu złożeniem wierszy (lub kolumn) macierzy C . Przykład takiego sposobu kodowania dla $n = 5$ ilustruje rysunek 7.38. Jeżeli n oznacza liczbę neuronów w sieci, to połączenia między tymi neuronami są przedstawione w postaci ciągu binarnego o długości n^2 . Oczywiście wadą takiego schematu kodowania jest gwałtowny wzrost długości genotypu wraz z powiększeniem się sieci neuronowej. Jednakże pewne ograniczenia mogą być łatwo dołączone do takiego schematu reprezentacji, co prowadzi do skrócenia długości chromosomów [137]. Można na przykład rozważać tylko połączenia jednokierunkowe, tzn. uwzględnić tylko te elementy macierzy C , które dotyczą połączenia danego węzła (neuronu) z następnym. Wtedy chromosom z przykładu z rysunku 7.38 będzie następujący: 0110110011.



Rys. 7.38. Przykład bezpośredniego kodowania macierzy połączeń dla sieci neuronowej

Kodowanie bezpośrednie ma swoje zalety, mianowicie dość łatwą możliwość oceny architektury sieci neuronowej oraz łatwość kodowania bądź dekodowania struktury. Jest to dość wygodne i skuteczne przy projektowaniu małych sieci neuronowych. W przypadku kodowania dużych sieci powstają jednak dłuższe chromosomy i zmniejsza się

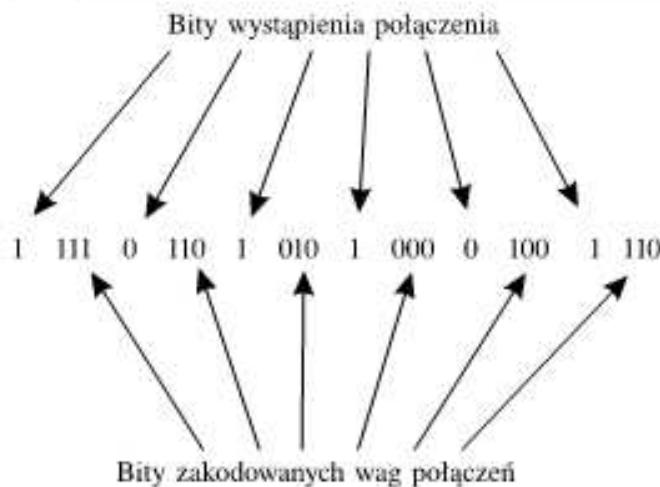
efektywność działania algorytmu ewolucyjnego. Wyjściem z sytuacji może być zastosowanie kodowania pośredniego, które polega na kodowaniu tylko najważniejszych cech zamiast każdego połączenia sieci neuronowej. Kodowanie to w ogólnym założeniu polega na poszukiwaniu użytecznych wspólnych bloków, które powtarzają się w konstrukcji sieci neuronowej. Reprezentacja ta wykorzystuje m.in. kodowanie używane w programowaniu genetycznym (punkt 7.3.5) lub też w innych metodach opartych na zapisywaniu grafów. Schemat ten jest bardziej uzasadniony biologicznie, gdyż zgodnie z odkryciami w dziedzinie neurologii nie jest możliwe, aby informacja genetyczna zakodowana w chromosomach określała cały system nerwowy bezpośrednio i niezależnie. Wynika to z faktu, że na przykład genotyp człowieka zawiera dużo mniej genów, niż ludzki mózg zawiera neuronów.

Drugi etap ewolucyjnego projektowania architektury sieci neuronowej przebiega, zgodnie z typowym cyklem ewolucji, według następujących kroków:

- 1) Dekodowanie każdego osobnika bieżącej generacji do architektury wynikającej z przyjętego schematu kodowania.
- 2) Uczenie każdej sieci neuronowej o architekturze otrzymanej w punkcie 1 za pomocą ustalonej wcześniej reguły uczenia (pewne parametry reguły uczenia mogą być adaptacyjnie aktualizowane podczas procesu uczenia). Uczenie należy rozpocząć od losowo wybranych początkowych wartości wag oraz parametrów reguły uczenia, jeśli takie występują.
- 3) Ocena przystosowania każdego osobnika (zakodowanej architektury) na podstawie powyższych rezultatów uczenia, tj. najmniejszego całkowitego średniego kwadratowego błędu uczenia, lub na podstawie testowania, jeżeli większy nacisk kładzie się na generalizację, najkrótszy czas uczenia, złożoność architektury (np. najmniejszą liczbę neuronów i połączeń między nimi) itp.
- 4) Reprodukcja osobników z prawdopodobieństwem odpowiednim do ich przystosowania lub rangi, w zależności od użytej metody selekcji.
- 5) Zastosowanie operatorów genetycznych, takich jak krzyżowanie, mutacja i/lub inwersja, i otrzymanie nowej generacji.

7.5.3. Algorytmy ewolucyjne do uczenia wag i określania topologii sieci neuronowej

Opisany w poprzednim punkcie proces ewolucji architektur cechuje się wieloma wadami. Uczenie wymaga długiego czasu obliczeniowego komputera, a jego wynik zależy od początkowego zainicjowania wag połączeń. Kolejną wadą jest tak zwany *problem permutacji*, polegający na tym, że jeden fenotyp może mieć dużo różnych odpowiedników w genotypach, tzn. daną sieć neuronową można różnoraką zapisać w chromosomie. Wady te można wyeliminować, łącząc metody kodowania wag i struktur (opartych na kodowaniu połączeń), które przedstawiono w punktach 7.5.1 i 7.5.2. Najprostszy sposób reprezentacji sieci neuronowej pokazaliśmy na rysunku 7.39. Połączenie neuronów



Rys. 7.39. Jednoczesne kodowanie struktury i wag sieci neuronowej

w sieci jest określone, podobnie jak w reprezentacji macierzowej, za pomocą jednego bitu. Dodatkowo wprowadzono zakodowane binarnie wartości wag połączeń. Alternatywnie można zastosować dwupoziomową reprezentację, oddzielając wartości wag od samych połączeń. Jedna część chromosomu będzie zawierała wzór połączeń, druga wartości wag. Oba sposoby charakteryzują się tym, że jeżeli gen odpowiadający za połączenie jest nieaktywny (przyjmuje wartość zero), kodowane wartości wag nie sąbrane pod uwagę przy dekodowaniu struktury.

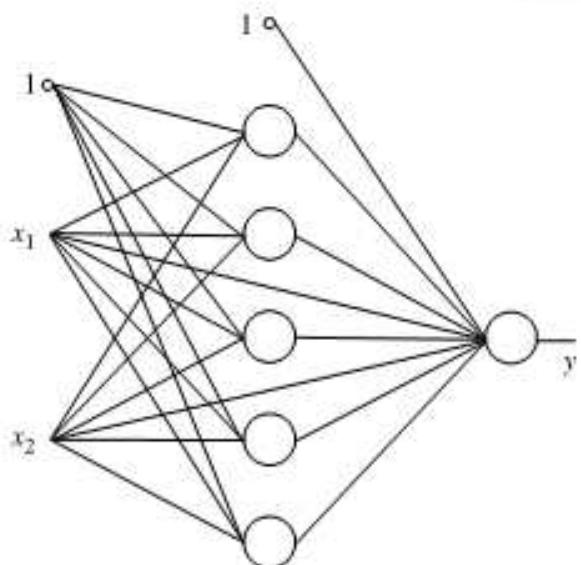
Inny sposób kodowania informacji o wagach i połączeniach sieci jednocześnie wykorzystuje reprezentację przedstawioną na rysunku 7.35. Informacja o istnieniu połączenia zawarta jest w samej wartości wagi. Jeżeli wartość ta wynosi zero, to połączenie tej wagi nie jestbrane pod uwagę. Metoda wymaga użycia dodatkowego operatora genetycznego, który z pewnym prawdopodobieństwem usuwałby bądź tworzył nowe połączenia (zerując bądź losując wartości wag połączeń). Cykl działania algorytmu ewolucyjnego stosowanego do jednoczesnego uczenia wag i określenia struktury sieci neuronowej jest podobny do opisanego w punkcie 7.5.2.

Przykład 7.25

Zademonstrujemy teraz przykład wykorzystania zmodyfikowanego algorytmu genetycznego do znalezienia odpowiedniej architektury i wartości wag połączeń sieci neuronowej. Sieć zostanie zastosowana do rozwiązywania zagadnienia XOR, a zatem mamy dwa wejścia i jedno wyjście. Szukamy sieci o strukturze warstw ($x; 1$), gdzie x oznacza nieznaną liczbę neuronów w warstwie pierwszej (w tym przypadku tożsamą z warstwą ukrytą). Zastosujemy kodowanie oparte na połączeniach sieci (punkt 7.5.2). Najpierw określmy maksymalny rozmiar sieci, by móc zakodować wszystkie możliwe połączenia pomiędzy neuronami. Przyjmujemy, że warstwa pierwsza może mieć co najwyżej 5 neuronów. Maksymalną strukturę sieci pokazano na rysunku 7.40. Przygotowanie algorytmu genetycznego rozpoczynamy od określenia sposobu kodowania zadania. Wagi będziemy kodować binarnie, każdą za pomocą 8 bitów $s(i) \in \{0, 1\}$, $i = 0, \dots, 7$, w sposób następujący:

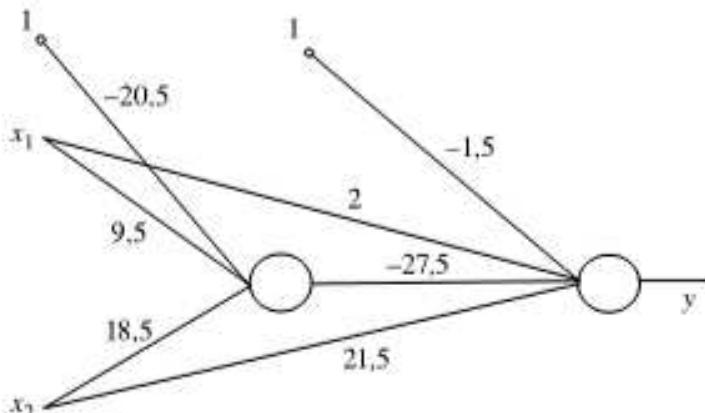
$$w = (-1)^{s(0)} \left(\sum_{i=1}^7 2^{i-2} s(i) \right).$$

Przyjmujemy pierwszy z bitów $s(0)$ jako znak wartości wagi. Zastosowanie powyższego



Rys. 7.40. Maksymalna architektura sieci neuronowej w przykładzie 7.25

wzor u prowadzi do ustalenia zakresu poszukiwań wartości wag od $-31,5$ do $31,5$. Jak łatwo sprawdzić, maksymalna struktura sieci ma 17 połączeń między neuronami i dodatkowo 6 połączeń, poprzez które podawana jest stała wartość. Zatem chromosom będzie reprezentował 23 wagi połączeń, z których każda jest kodowana za pomocą 8 bitów, co daje 184 geny. Do krzyżowania stosujemy operator dwupunktowy, natomiast rozróżniamy dwa operatory mutacji. Działanie pierwszego z nich polega na wylosowaniu nowej wartości wagi. Drugi operator mutacji będzie odpowiedzialny za zmianę struktury sieci, zakładamy bowiem, że połączenie istnieje wówczas, gdy wartość wagi jest różna od zera. Gdy wszystkie bity kodujące wagę mają wartość równą zero, wówczas dane połączenie nie jest brane pod uwagę. W ten sposób brak połączeń wejściowych lub wyjściowych w pewnym neuronie prowadzi do pominięcia tego neuronu w strukturze sieci. Dzięki temu modyfikuje się architektura sieci neuronowej. Działanie drugiego operatora mutacji będzie polegać na usunięciu (przyjmujemy wartości wagi równe 0) bądź dodaniu połączenia (losujemy wartości 8 genów). Zadaniem funkcji przystosowania jest ocena sieci. Bierzymy pod uwagę rozmiar sieci, przy czym duże znaczenie ma zmniejszenie liczby neuronów. Ponadto ważnym wskaźnikiem jest błąd średni kwadratowy, jaki uzyskamy przy testowaniu sieci. Wynik działania algorytmu ilustruje rysunek 7.41, na którym pokazano sieć, jaką uzyskano po 211 generacjach. Populacja liczyła 100 osobników. Jak widać, struktura sieci jest nieco inna od przedstawionej w przykładzie 7.24, składa się bowiem tylko z dwóch neuronów. Błąd średni kwadratowy uzyskany dla tej sieci wynosił 0,0372. Sieć można jeszcze douczyć za pomocą algorytmu wstępnej propagacji błędów.



Rys. 7.41. Architektura sieci neuronowej realizującej funkcję XOR, znaleziona przez algorytm ewolucyjny przedstawiony w przykładzie 7.25.

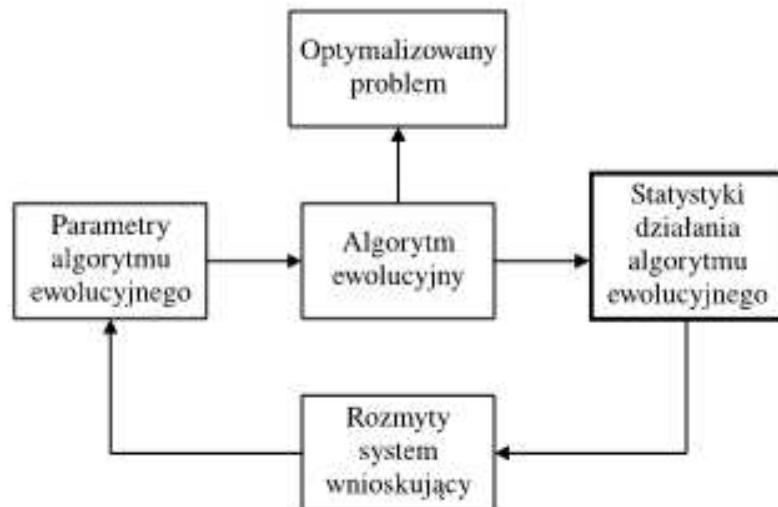
7.6. Algorytmy ewolucyjne a systemy rozmyte

W literaturze opisano różne sposoby połączenia algorytmów ewolucyjnych i systemów rozmytych. Jedna z możliwości polega na kontroli pracy algorytmu ewolucyjnego z wykorzystaniem rozmytej bazy wiedzy. Takie połączenie pozwala sterować parametrami algorytmu oraz monitorować jego pracę w celu uniknięcia niepożądanych zachowań, takich jak przedwczesna zbieżność. Inną opcją jest wykorzystanie logiki rozmytej w samym algorytmie ewolucyjnym; istnieje możliwość zdefiniowania rozmytych operacji genetycznych, a nawet rozmytych genów. Jedna z najczęściej opisywanych metod hybrydowych polega na wykorzystaniu algorytmów ewolucyjnych do optymalizacji systemów rozmytych.

7.6.1. Systemy rozmyte do kontroli ewolucji

W punkcie 7.4.1 zwróciliśmy uwagę na problem eksploracji i eksploracji algorytmu ewolucyjnego. Jest szczególnie ważne, by algorytm podczas swojego działania był odpowiednio wyważony pomiędzy tymi dwoma skrajnymi przypadkami. Jeśli na przykład wszystkie osobniki są do siebie podobne, a ciągle jesteśmy daleko od optymalnego rozwiązania, należałoby nastawić się na większą eksplorację przestrzeni poszukiwań. I odwrotnie, zbytnia różnorodność populacji powoduje niemożność znalezienia optimum. Można więc na przykład zwiększyć nieco nacisk selektywny, powodując wzrost eksploracji. Jest to jedna z możliwości wpływania na pracę algorytmu ewolucyjnego. Oprócz nacisku selektywnego, możemy zmieniać prawdopodobieństwa krzyżowania i mutacji. Ekspert podczas pracy algorytmu sprawdza jego zachowanie. Na podstawie odpowiednio dobranych statystyk możemy zmieniać parametry algorytmu tak, by uniknąć przedwczesnej zbieżności. Ponadto możemy decydować o jakości rozwiązania w zależności od czasu działania algorytmu.

Alternatywą do ingerencji eksperta w działanie algorytmu są tzw. *adaptacyjne algorytmy ewolucyjne*, zdolne samodzielnie nastawiać swoje parametry podczas pracy w celu osiągnięcia jak najlepszych rezultatów. Do modyfikacji tych parametrów wykorzystuje się rozmyte systemy wnioskujące. Ogólny schemat adaptacyjnego algorytmu ewolucyjnego przedstawiono na rysunku 7.42.



Rys. 7.42. Schemat blokowy adaptacyjnego algorytmu ewolucyjnego

Podczas pracy algorytmu ewolucyjnego generowane są różne statystyki. Na podstawie tych informacji rozmyty system wnioskujący sprawdza poprawność pracy algorytmu i dynamicznie wpływa na jego działanie poprzez zmianę niektórych parametrów (np. prawdopodobieństwo mutacji). Rozmyty system wnioskujący ma zaimplementowaną bazę wiedzy, odpowiednio przygotowaną przez eksperta w formie reguł lingwistycznych. Informacje dotyczące pracy algorytmu (statystyki) są podawane na wejście systemu, podlegają rozmywaniu, czyli są zamieniane na zbiory rozmyte. Rezultatem działania systemu rozmytego może być podjęcie decyzji o zmianie któregoś z parametrów algorytmu ewolucyjnego. Kontrolowanymi parametrami mogą być prawdopodobieństwa mutacji i krzyżowania, rozmiar populacji, nacisk selektywny lub też inne parametry zależne od rozwiązywanego problemu. Kontrola ta odbywa się na bieżąco, niejako „w locie”. Automatycznie steruje się wówczas pracą algorytmu tak, by był zachowany odpowiedni stosunek między eksploracją a eksploatacją.

Statystyki algorytmu ewolucyjnego, na podstawie których sterownik rozmyty podejmuje decyzje, są sprawdzane co pewną ustaloną liczbę generacji. Statystyki te można podzielić na dwie grupy. Pierwsza dotyczy relacji między genotypami osobników w populacji, np. badanie różnorodności osobników za pomocą pewnej funkcji miary. Druga zajmuje się badaniem przystosowania fenotypów. Jako miarę można przyjmować maksymalną, średnią i minimalną wartość funkcji przystosowania w populacji lub stosunek najlepszej do średniej wartości przystosowania. Można też zastosować zupełnie inne statystyki, na przykład liczbę mutacji, które poprawiły przystosowanie, w stosunku do liczb wszystkich mutacji (prowadzi to do tych samych obliczeń, co w regule 1/5 sukcesów — punkt 7.3.3).

Przykład 7.26

Pokażemy przykład ilustrujący, jak wygląda baza reguł, za pomocą której można sterować parametrami algorytmu ewolucyjnego. Danymi wejściowymi dla sterownika rozmytego będą cztery parametry; dwa z nich to dane statystyczne odzwierciedlające zachowanie się algorytmu, czyli stosunek średniej do najlepszej i najgorszej do średniej wartości funkcji przystosowania. Dodatkowo bierzemy pod uwagę prawdopodobieństwo krzyżowania p_k i wielkość populacji. Do bazy reguł wprowadzamy wartości lingwistyczne: *duże*, *średnie* i *małe*. Oznaczmy

$$\alpha = \frac{\text{średnie przystosowanie}}{\text{najlepsze przystosowanie}},$$

$$\beta = \frac{\text{najgorsze przystosowanie}}{\text{średnie przystosowanie}}.$$

Przykładowy fragment bazy reguł może wyglądać następująco [125]:

JEŻELI α jest *duże* **TO** zwiększa populację

JEŻELI β jest *małe* **TO** zmniejsza populację

JEŻELI p_k jest *małe* **I** populacja jest *mała* **TO** zwiększa populację

Widzimy, że sterowanym parametrem jest wielkość populacji, poprzez którą możemy wpływać na różnorodność osobników.

7.6.2. Ewolucja systemów rozmytych

Projektując system rozmyty, stojmy przed zadaniem doboru odpowiedniej bazy reguł. Podobnie jak w przypadku projektowania sieci neuronowej, wymaga to czasu, doświadczenia i wiedzy eksperta. Proces ten można zautomatyzować dzięki algorytmom ewolucyjnym, których elastyczność i niezależność od rozwiązywanego problemu wykorzystujemy między innymi w następujących momentach projektowania systemu rozmytego:

- Gdy dysponujemy systemem rozmytym i dążymy do poprawienia skuteczności jego działania. Algorytmy ewolucyjne wykorzystywane mogą być do dopasowania (ang. *tuning*) funkcji przynależności, tzn. zmiany ich położenia lub kształtu.
- Gdy określony jest zbiór funkcji przynależności określeń lingwistycznych, możemy metodami ewolucyjnymi wygenerować bazę reguł. Stosuje się trzy podejścia do rozwiązania tego problemu — Michigan, Pittsburgh i uczenie iteracyjne, wyróżniające się sposobem kodowania i konstruowania bazy reguł.

Podejście Michigan, wykorzystuje ideę tzw. *systemów klasyfikatorowych*. Każdy z osobników reprezentuje jedną zakodowaną regułę. Wszystkie lub też tylko część chromosomów populacji traktowana jest jako poszukiwana baza reguł.

Podejście Pittsburgh jest sposobem kodowania bardziej odpowiadającym działaniu algorytmów ewolucyjnych. Cała baza reguł zakodowana jest w jednym chromosomie. Poszukiwane rozwiązanie znajdujemy więc w najlepiej przystosowanym osobniku.

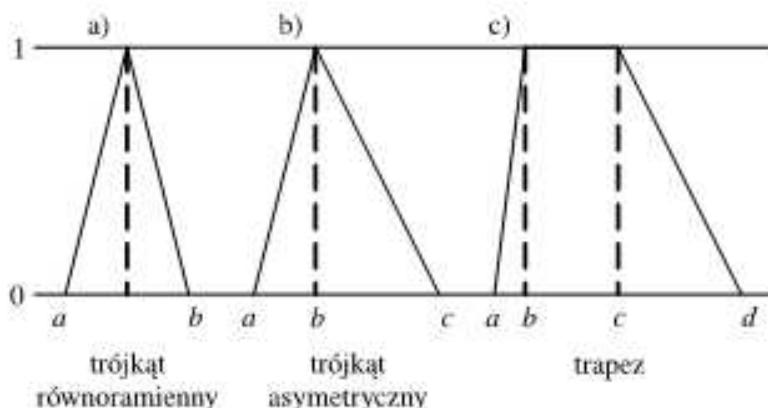
Podejście trzecie poszukiwania bazy reguł to tzw. *uczenie iteracyjne* (ang. *iterative rule learning*). Łączy ono najlepsze cechy podejścia Michigan i Pittsburgh. Wykorzystano pomysł kodowania jednej reguły w chromosomie, ale utworzenie całej bazy odbywa się stopniowo. Ostateczną bazę reguł tworzą najlepsze osobniki, będące rezultatem działania kolejnych uruchomień algorytmu ewolucyjnego.

Omówimy sposoby ewolucyjnego dopasowywania funkcji przynależności, a w szczególności metody kodowania tych funkcji. Następnie przedstawimy ideę podejścia Michigan, Pittsburgh oraz uczenia iteracyjnego.

7.6.2.1. Dopasowanie funkcji przynależności

Skuteczność działania systemu rozmytego można zwiększyć przez odpowiednie dopasowanie zbiorów rozmytych, nie zmieniając przy tym bazy reguł. Algorytm ewolucyjny modyfikuje funkcje przynależności poprzez zmianę położenia punktów charakterystycznych ich kształtów. Są to najczęściej współrzędne wierzchołków figur, które te funkcje opisują. Informacje o wierzchołkach kodowane są w chromosomach. Kształt zbiorów i kodowanie są ze sobą ściśle powiązane. Można wyróżnić kilka popularnych funkcji (rys. 7.43):

- trójkąt równoramienny — funkcje zakodowane są w chromosomie za pomocą dwóch skrajnych punktów trójkąta a i b (rys. 7.43a).
- trójkąt asymetryczny — położenie i kształt trójkąta asymetrycznego kodujemy trzema parametrami a , b i c (rys. 7.43b). W porównaniu z trójkątem równoramiennym dodatkowo określa się położenie środkowego wierzchołka. Można również zamiast współrzędnych dwóch skrajnych punktów podać odległości od środkowego punktu.



Rys. 7.43. Różne kształty funkcji przynależności z zaznaczonymi charakterystycznymi wierzchołkami

• trapez — trapez charakteryzuje cztery punkty (rys. 7.43c), ale należy pamiętać o tym, podobnie zresztą jak w przypadku trójkątów, aby punkty a , b , c i d , które reprezentują trapez, spełniały warunek $a < b < c < d$.

• można wyróżnić również inne funkcje, np. funkcję gaussowską, którą można opisać za pomocą dwóch parametrów: środka \bar{x} oraz szerokości σ . Inne funkcje, które są stosowane, to funkcja radialna lub sigmoidalna (dla skrajnych zbiorów rozmytych, reprezentujących wartości zmiennych lingwistycznych).

W zależności od zastosowanego algorytmu ewolucyjnego, punkty charakterystyczne funkcji przynależności koduje się w chromosomie w postaci binarnej lub za pomocą liczb rzeczywistych. Po wybraniu odpowiedniej reprezentacji zbiorów rozmytych w chromosomie, algorytm ewolucyjny działa na populacji osobników (chromosomów zawierających zakodowane kształty funkcji przynależności systemu rozmytego) według cyklu ewolucyjnego, który może obejmować następujące kroki:

1) Dekodowanie każdego z osobników populacji polega na odtworzeniu zbioru funkcji przynależności i skonstruowaniu odpowiedniego systemu rozmytego. Baza reguł jest wcześniej ustalona.

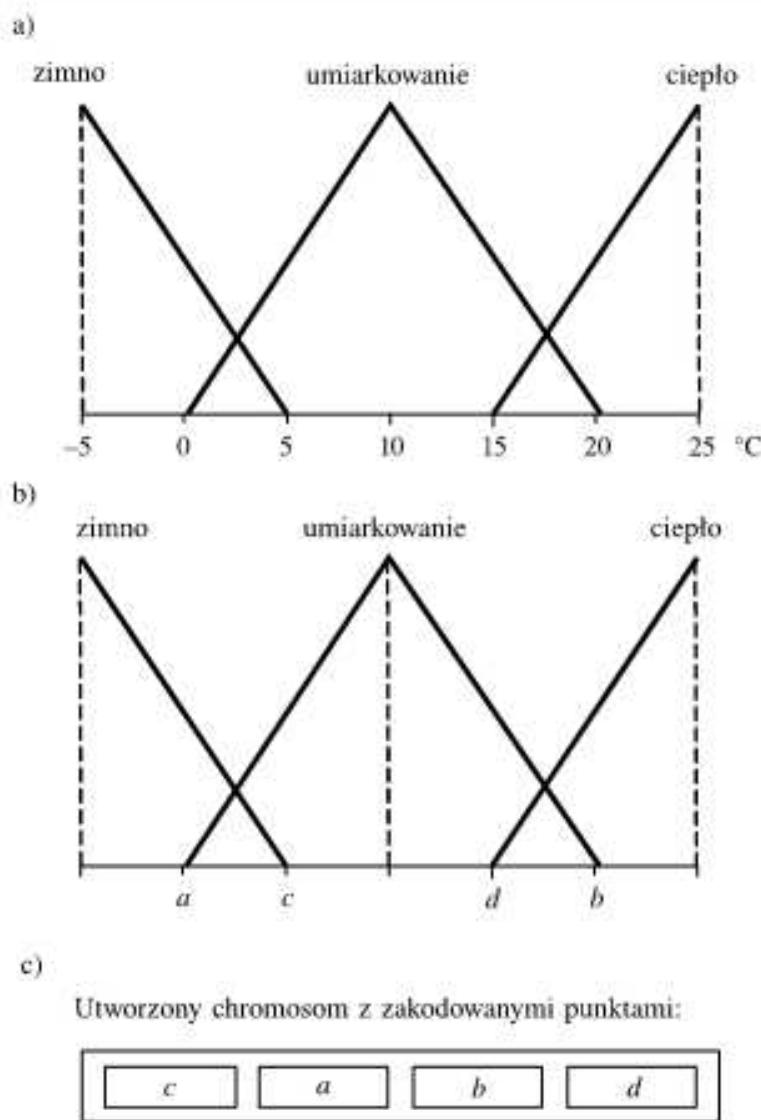
2) Oceny pracy systemu rozmytego dokonuje się na podstawie różnicy (błędu) między odpowiedziami systemu a wartościami wzorcowymi. Błąd ten określa przystosowanie osobnika.

3) Reprodukcja osobników i zastosowanie operatorów genetycznych. Konkretnie techniki zależą od wybranego algorytmu ewolucyjnego, gdyż algorytm genetyczny i strategie ewolucyjne cechują się różnymi mechanizmami selekcji i rekombinacji.

4) Jeżeli warunek zatrzymania nie jest spełniony, przechodzimy do punktu 1.

Przykład 7.27

Rozpatrzmy przykładowe zbiory rozmyte (wartości lingwistyczne) *zimno*, *umiarkowanie* i *ciepło*, które mogą posłużyć do opisu temperatury otoczenia (rys. 7.44a). Początkowo zakładamy, że określenie *zimno* dotyczy temperatury poniżej 5°C , *umiarkowanie* — gdy temperatura zmienia się w granicach od 0 do 20°C , natomiast *ciepło* — gdy temperatura przekracza 15°C . Wierzchołek odpowiadający temperaturze -5°C (zimno) oraz temperaturze 25°C (ciepło) nie podlega ewolucyjnym zmianom. Funkcje przynależności mają kształt trójkątów, które można opisać za pomocą punktów charakterystycznych w następujący sposób (rys. 7.44b): dwa skrajne zbiory za pomocą jednego punktu (*zimno* — c , *ciepło*



Rys. 7.44. Przykład kodowania trójkątnych funkcji przynależności

— d , pozostałe wierzchołki trójkąta są stałe), natomiast środkowy zbiór *umiarkowanie* — dwoma punktami (a i b , wierzchołek trójkąta znajduje się w połowie odległości między tymi punktami). Spróbujmy zakodować te zbiory rozmyte w chromosomie, umieszczając punkty charakterystyczne kolejno obok siebie (rys. 7.44c). Pierwszy z nich dotyczy wartości lingwistycznej *zimno* i określony jest jedną wartością c , zbiór rozmyty *umiarkowanie* dwoma punktami a i b , natomiast zbiór rozmyty *ciepło* punktem d . Wartości te można przedstawić jako liczby rzeczywiste bądź łańcuchy binarne, stosując jedną z poznanych metod kodowania (punkt 7.4.5).

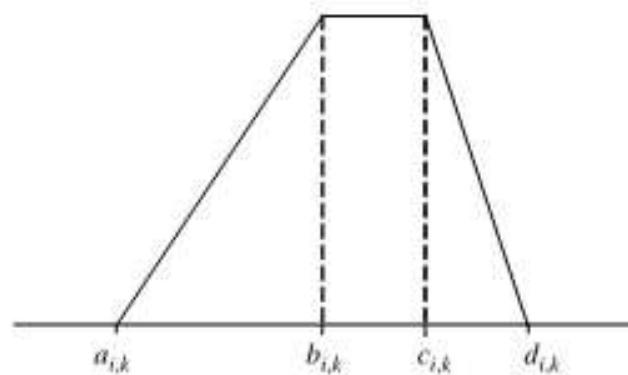
Przykład 7.28

W przykładzie tym pokażemy, jak zakodować w chromosomie system rozmyty typu Mamdaniego o N regułach i n wejściach, z trapezowymi funkcjami przynależności. Rozważmy k -tą regułę postaci (podrozdz. 4.9)

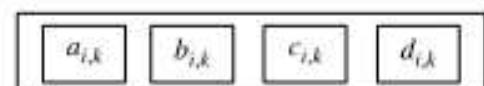
$R^{(k)}$: **JEŻELI** x_1 jest A_1^k **I** x_2 jest A_2^k ... **I** x_n jest A_n^k **TO** y jest B^k

Wszystkie zbiory rozmyte A_i^k , $i = 1, \dots, n$, $k = 1, \dots, N$, są kodowane, jak pokazano na rysunku 7.45. Zbiory te są reprezentowane poprzez następujące chromosomy:

$$C_{i,k} = (a_{i,k}, b_{i,k}, c_{i,k}, d_{i,k}).$$



Chromosom z zakodowanymi punktami:



Rys. 7.45. Przykład kodowania funkcji przynależności o kształcie trapezu

Uwzględniając teraz wszystkie zbiory rozmyte w k -tej regule, otrzymujemy

$$C_k = (a_{1,k}, b_{1,k}, c_{1,k}, d_{1,k}, \dots, a_{n,k}, b_{n,k}, c_{n,k}, d_{n,k}, a'_k, b'_k, c'_k, d'_k),$$

przy czym zbiór B^k jest również zbiorem trapezowym, opisanym za pomocą punktów (a'_k, b'_k, c'_k, d'_k) . Ostateczny chromosom C , kodujący wszystkie reguły, uzyskamy przez połączenie kolejnych fragmentów C_k , $k = 1, \dots, N$, tzn.

$$C = (C_1, C_2, \dots, C_N).$$

Chromosom ten ma długość równą $4N(n + 1)$. Dlatego też przy większej liczbie reguł warto zastosować kodowanie z wykorzystaniem liczb rzeczywistych.

Przykład 7.29

Zademonstrujemy zakodowanie w chromosomie reguł systemu rozmytego typu Takagi–Sugeno (rozdział 9). Zakładamy, że system ma N reguł oraz n wejścia, natomiast funkcje przynależności poszczególnych zbiorów rozmytych są funkcjami trapezowymi. Poniżej jest przedstawiona k -ta reguła tego systemu

$$R^{(k)} : \text{JEŻELI } x_1 \text{ jest } A_1^k \dots \text{ I } x_n \text{ jest } A_n^k \text{ TO } y = c_0^{(k)} + c_1^{(k)}x_1 + \dots + c_n^{(k)}x_n$$

Jak widzimy, w następnikach reguł występują zależności funkcyjne (funkcje liniowe) z parametrami $c_j^{(k)}$, $j = 0, \dots, n$, $k = 1, \dots, N$. Owe parametry musimy również zakodować w chromosomie. We fragmencie osobnika opisującego tę regułę można wyodrębnić dwie części. Pierwsza z nich opisuje współrzędne wszystkich funkcji przynależności poprzedników reguł i wygląda następująco:

$$C_k^1 = (a_{1,k}, b_{1,k}, c_{1,k}, d_{1,k}, \dots, a_{i,k}, b_{i,k}, c_{i,k}, d_{i,k}, \dots, a_{n,k}, b_{n,k}, c_{n,k}, d_{n,k}).$$

W drugiej części umieścimy parametry nastęników reguł

$$C_k^2 = (c_0^{(k)}, c_1^{(k)}, \dots, c_n^{(k)}).$$

Kodując parametry $c_j^{(k)}$, musimy znać zakres wartości, które przyjmują, co jest nieodzowne choćby dla kodowania binarnego. Niestety, najczęściej nie znamy tego zakresu. W tym przypadku można zastosować metodę znaną w literaturze angielskojęzycznej pod nazwą

angular coding [28]. Zamiast bezpośredniego kodowania parametrów $c_j^{(k)}$ można zakodować wartości $\alpha_{j,k} = \arctan c_j^{(k)}$, które znajdują się w przedziale $(-\frac{\pi}{2}, \frac{\pi}{2})$. Pozwala to na wybór dowolnej metody kodowania. Podobnie jak w poprzednim przykładzie, cały chromosom składa się z fragmentów zakodowanych powyżej reguł. Chromosom z N regułami mógłby wyglądać tak

$$C = (C_1^1, C_2^1, \dots, C_N^1, C_1^2, C_2^2, \dots, C_N^2).$$

7.6.2.2. Ewolucja reguł

Jak już wspomniano we wstępie podrozdziału, można rozróżnić trzy metody wykorzystania algorytmów ewolucyjnych w procesie generacji reguł systemu rozmytego: podejście Michigan, podejście Pittsburgh, iteracyjny proces uczenia (ang. *iterative learning process*). Przyjrzyjmy się kolejno poszczególnym metodom.

7.6.2.2.1. Podejście Michigan

Podejście to zostało opracowane na Uniwersytecie Michigan. Cechą charakterystyczną tego podejścia jest to, że poszczególne reguły są kodowane w osobnych chromosomach. Oryginalna metoda Michigan korzysta z koncepcji tzw. systemu klasyfikatorowego. Zainteresowanego czytelnika odsyłamy do literatury na ten temat [29, 35, 63]. Obecnie przedstawimy jedynie sposób kodowania poszczególnych reguł w osobnych chromosomach.

Przykład 7.30

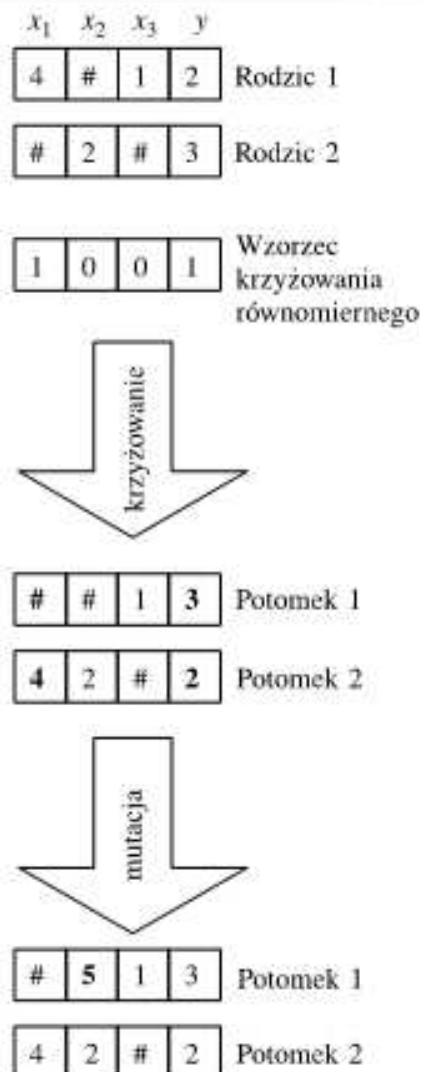
Rozważmy system rozmyty z trzema wejściami i jednym wyjściem. Zmienne lingwistyczne przyjmują następujące wartości:

- 1: *bardzo mało*,
- 2: *mało*,
- 3: *średnio*,
- 4: *dużo*,
- 5: *bardzo dużo*.

Każde z określeń będzie kodowane odpowiadającą mu cyfrą. Dodajmy jeszcze znak oznaczający brak odpowiedniej wartości lingwistycznej: #. Na rysunku 7.46 zakodowano dwie reguły (Rodzic 1 i Rodzic 2). Na przykład chromosom 4#12 oznacza następującą regułę:

JEŻELI x_1 jest *dużo* **I** x_3 jest *bardzo mało* **TO** y jest *mało*

Zmienna x_2 , jak widać, nie występuje, gdyż w odpowiednim miejscu chromosomu umieszczony jest znak #. Generowanie nowych osobników odbywa się za pomocą znanych nam operatorów genetycznych — krzyżowania i mutacji. Przykładowe operacje przedstawia rysunek 7.46. Zastosowano tu krzyżowanie jednorodne oraz mutację polegającą na wylosowaniu nowej wartości genu. Selekcja polega na odrzuceniu osobników najgorzej przystosowanych, a na ich miejsce wchodzą nowe, utworzone za pomocą operatorów genetycznych. W literaturze można spotkać wiele metod automatycznego generowania bazy reguł, które określane są jako podejście Michigan. Większość z nich nie ma nic wspólnego z klasycznym algorytmem ewolucyjnym, ale jest to dobre miejsce na przedstawienie tego podejścia.



Rys. 7.46. Operacje genetyczne przeprowadzane na chromosomach w podejściu Michigan

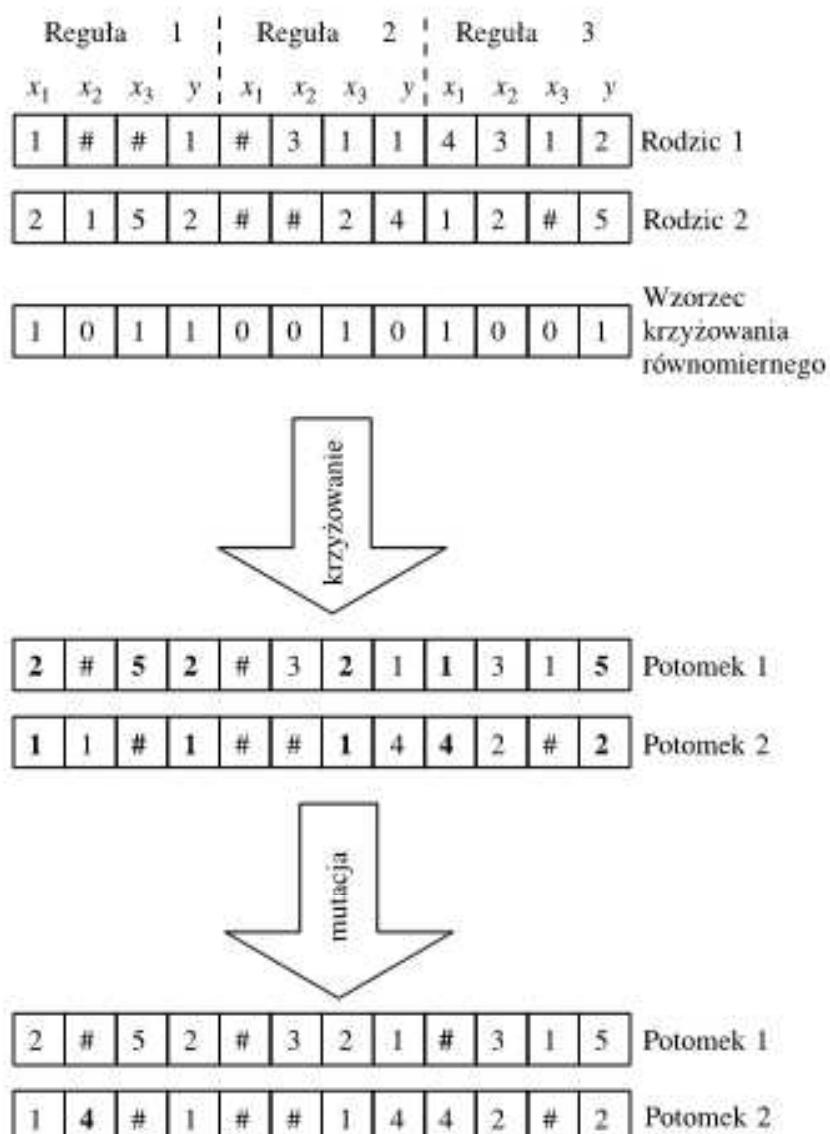
nego z algorytmem pracy systemów klasyfikatorowych, z których wywodzi się oryginalna metoda Michigan. Łączy je jednak pomysł reprezentacji jednej reguły w chromosomie oraz traktowanie całej populacji jako poszukiwanego zbioru reguł.

7.6.2.2.2. Podejście Pittsburgh

Podejście to zapoczątkowano na Uniwersytecie w Pittsburghu i stąd też pochodzi nazwa. Cechą charakterystyczną tej metody jest umieszczenie całego rozwiązania w pojedynczym chromosomie. Każdy osobnik reprezentuje kompletny zbiór reguł. Osobniki rywalizują ze sobą, słabe wymierają, mocne przeżywają i reprodukują się. Dokonuje się tego na podstawie selekcji oraz operatorów krzyżowania i mutacji. Możliwe jest zachowanie równowagi pomiędzy eksploracją nowych rozwiązań a eksploatacją najlepszego rozwiązania. Działanie podejścia Pittsburgh nie odbiega od działania klasycznego algorytmu genetycznego. Niestety są i wady. Chromosom kodujący wszystkie reguły jest dużo większy niż w metodzie Michigan, przez co wydłuża się czas pracy algorytmu ewolucyjnego, a także maleje szansa znalezienia poprawnego rozwiązania. W najprostszym algorytmie poszukiwania rozmytej bazy reguł należy odgórnie ustalić maksymalną liczbę reguł. Zbiór wszystkich reguł systemu rozmytego przechowywany jest w postaci stałej długości łańcucha z rozgraniczeniem na podłańcuchy, w których zakodowane są poszczególne reguły.

Przykład 7.31

Pokażemy sposób kodowania w metodzie Pittsburgh, rozważając system rozmyty opisany w przykładzie 7.30. Na rysunku 7.47 zakodowano dwie bazy, z których każda składa



Rys. 7.47. Operacje genetyczne przeprowadzane na chromosomach w podejściu Pittsburgh

się z 3 reguł (Rodzic 1 i Rodzic 2). Analogicznie jak w przykładzie 7.30, chromosom odpowiadający Rodzicowi 1 można rozkodować następująco:

$R^{(1)}$: **JEŻELI** x_1 jest bardzo mało **TO** y jest bardzo mało

$R^{(2)}$: **JEŻELI** x_2 jest średnio I x_3 jest bardzo mało **TO** y jest bardzo mało

$R^{(3)}$: **JEŻELI** x_1 jest dużo I x_2 jest średnio I x_3 jest bardzo mało **TO** y jest mało

Działanie operatorów genetycznych przedstawia rysunek 7.47. Operatory te działają podobnie jak w metodzie Michigan (rys. 7.46), ale wyraźnie widać różnicę w długości chromosomu. Oczywiście jest to bardzo prosty algorytm generacji reguł, jednak można zastosować jego liczne modyfikacje, jak choćby zmienną długość chromosomu, do generacji różnej liczby reguł. Dodatkowo, kodując kształty funkcji przynależności w sposób

omówiony w punkcie 7.6.2.1, oprócz generacji reguł możemy jednocześnie dostrajając parametry funkcji przynależności poszczególnych zbiorów rozmytych.

7.6.2.2.3. Uczenie iteracyjne

Uczenie iteracyjne powstało jako próba kombinacji najlepszych cech poprzednio opisanych podejść Michigan i Pittsburgh. Metoda ta charakteryzuje się tym, że każdy chromosom w populacji reprezentuje pojedynczą regułę, podobnie jak w podejściu Michigan. Charakterystyczne dla podejścia Pittsburgh jest natomiast to, że wybierany jest tylko najlepszy z osobników populacji. Pełną bazę reguł uzyskujemy poprzez wielokrotne powtarzanie algorytmu ewolucyjnego, za każdym razem dodając do bazy reguł jedną najlepszą regułę, aż do znalezienia całego rozwiązania. Algorytm generacji bazy reguł oparty na uczeniu iteracyjnym można przedstawić następująco:

- 1) Zastosuj algorytm ewolucyjny do znalezienia reguły. Jako przystosowanie chromosomu kodującego reguły można przyjąć np. liczbę poprawnie sklasyfikowanych danych uczących lub prostotę reguły.
- 2) Dołącz regułę do ostatecznego zbioru reguł.
- 3) Oceń skuteczność zbioru reguł ze szczególnym uwzględnieniem nowo znalezionej.
- 4) Jeżeli zbiór wygenerowanych reguł jest adekwatny do rozwiązywanego problemu, to zakończ działanie tego algorytmu. W przeciwnym wypadku przejdź do punktu 1.

Generowanie reguł w metodach Pittsburgh oraz Michigan polegało na równoczesnym otrzymaniu całej bazy reguł. W uczeniu iteracyjnym poszczególne reguły powstają niezależnie od siebie, bez jakiegokolwiek informacji o poprzednio wygenerowanych. Może to powodować powtarzanie się reguł bądź wzajemne ich wykluczanie. Dlatego często stosuje się, po zakończeniu pracy algorytmu, procedury uproszczenia bazy reguł. Można jednak już w trakcie pracy algorytmu zapobiegać pojawianiu się identycznych reguł. Metodą tą jest usuwanie z ciągu uczącego tych danych, które posłużyły do prawidłowego nauczenia poprzednio znalezionych reguł.

7.7. Uwagi

Pionierem i twórcą klasycznych algorytmów genetycznych był Alex Fraser, który pierwszą pracę [57] na ten temat opublikował w 1957 roku. Jego działalność naukowa jest niemal zupełnie nieznana w literaturze światowej. Goldberg [63], Holland [82] oraz Michalewicz [136] są autorami znanych monografii na temat algorytmów ewolucyjnych. W Polsce godna polecenia jest monografia Arabasa [2]. W monografiach [59, 60, 248, 260] autorzy poruszają różne zagadnienia dotyczące algorytmów ewolucyjnych oraz ich zastosowań. Popularnym programem do symulacji algorytmów genetycznych jest program Evolver [50]. W monografiach [2] i [136] autorzy opisują historyczne prace Ingo Rechenberga i Hansa-Paula Schwefela na temat strategii ewolucyjnych oraz Lawrence'a Fogla na temat programowania ewolucyjnego. Twórcą koncepcji programowania genetycznego jest John Koza [120]. W monografiach [24, 63, 82, 136] oraz pracy [175] poruszane jest twierdzenie

o schematach, hipoteza cegiełek (lub bloków budujących), a także teoretyczne podstawy działania algorytmów genetycznych. Różne algorytmy selekcji oraz skalowania opisano w pracach [38] oraz [75]. Zalety kodowania Graya opisano w monografiach [2] oraz [31]. Szczegółowy opis metod krzyżowania zamieszczono w opracowaniach [35] i [63].

Ewolacyjne uczenie małych jednokierunkowych sieci neuronowych opisano w pracy [256], a wykorzystanie algorytmu ewolucyjnego do uczenia względnie dużych sieci przedstawiono w pracy [230]. Opis ewolucyjnego projektowania architektury sieci neuronowej można znaleźć w pracach [232, 263, 264]. Różne metody kodowania wag sieci neuronowych zostały zaproponowane w pracach [6, 7, 139]. Metody kodowania bezpośredniego struktury sieci neuronowych omówiono w pracy [112]. Prace [110, 230, 263] podają sposoby kodowania pośredniego sieci neuronowych z wykorzystaniem grafów. Jednoczesne kodowanie wag i połączeń sieci neuronowych przedstawiono w pracach [15, 250, 251]. W pracy [22] opisano zastosowanie systemów rozmytych do kontroli ewolucji. Zagadnienie wykorzystania logiki rozmytej w algorytmach ewolucyjnych opisano w pracach [77] i [78]. Metody kodowania różnych funkcji przynależności podano w pracach [29, 102, 103]. Podejście Michigan wraz z opisem systemów klasyfikatorowych przedstawiono w monografii [29] oraz pracy [249]. Metody kodowania bazy reguł z wykorzystaniem podejścia Michigan i Pittsburgh opisano w pracy [88]. Szczegółowy opis uczenia iteracyjnego można znaleźć w monografii [29]. W pracach [36, 37, 182] autorzy omawiają zastosowania algorytmów ewolucyjnych do projektowania maszyn elektrycznych, w opracowaniu [75] do optymalizacji sieci elektroenergetycznych, w pracy [160] do planowania produkcji, natomiast w monografii [153] do diagnostyki technicznej.

Różnice pomiędzy omówionymi w tym rozdziale algorytmami ewolucyjnymi coraz bardziej się zacierają. Algorytmy te w swoich pierwotnych postaciach są obecnie rzadko spotykane. Stosuje się je najczęściej do testów porównawczych. Należy dodać, że nazwy „algorytmy genetyczne” używa się, rozumiejąc ją zarówno w węższym sensie, a więc jako klasyczne algorytmy genetyczne lub ich nieznaczne modyfikacje, ale też w sensie szerszym, mając na myśli algorytmy ewolucyjne znacznie odbiegające od klasycznego algorytmu genetycznego.

Na zakończenie tego rozdziału w tabeli 7.13 porównujemy najważniejsze cechy algorytmów ewolucyjnych, takie jak: metody kodowania, sposoby krzyżowania i mutacji oraz ich znaczenie w przebiegu algorytmu, metody selekcji, a także sposób tworzenia nowej populacji. Przedstawiliśmy potencjalne zastosowania oraz nazwiska twórców różnych algorytmów. Załączone zestawienie może być pomocą w wyborze odpowiedniego typu algorytmu do rozwiązania konkretnego problemu.

Tabela 7.13. Zestawienie głównych cech algorytmów ewolucyjnych

	Algorytm genetyczny (GA)	Strategie ewolucyjne (SE)	Programowanie
Kodowanie, reprezentacja	W klasycznym GA występuje kodowanie binarne. Można je zastąpić kodowaniem Graya lub zastosować inne kodowanie o ustalonym alfabetie	Reprezentacja za pomocą wektorów liczb zmiennoprzecinkowych, składających się z wartości zmiennych rozwiązywanego zadania i informacji o zasięgu mutacji	Reprezentacja odpowiadająca kodowanemu podobna do reprezentacji GA
Operator krzyżowania (rekombinacji)	Jednopunktowy, wielopunktowy (w szczególności dwupunktowy), jednorodny	Nie występuje w klasycznej wersji. Możliwe jest krzyżowanie uśredniające bądź wymieniające wartości elementów wektorów macierzystych	Brak
Rola krzyżowania	Pierwszoplanowa, p_k rzędu 0,5–1 oraz $p_k \gg p_m$	Drugoplanowa, możliwy brak krzyżowania	Brak
Operator mutacji	Zamiana wartości genów na przeciwną	Najpierw następuje losowa modyfikacja wartości odchyleń standardowych osobnika, a następnie modyfikacja wartości zmiennych niezależnych	Podobna jak w SE, samoczynna adaptacyjna do mutacji
Rola mutacji	Drugoplanowa, p_m rzędu 0–0,1	Pierwszoplanowa, stosowana jest autoadaptacja zasięgu mutacji	Jedyny operator genetyczny
Dobór rodziców lub tworzenie tymczasowej populacji	Prawdopodobieństwo wyboru osobnika do puli rodzicielskiej zależy od wartości jego funkcji przystosowania. Stosuje się metodę koła ruletki, turniejową, rankingową i inną	Tworzona jest tymczasowa populacja (o liczności λ) za pomocą losowania ze zwracaniem spośród μ osobników, której osobniki poddawane są operatorom genetycznym	Każdy z rodziców tworzy potomka, wykorzystując mutację
Tworzenie nowej populacji	Wszystkie wybrane osobniki w procesie selekcji są poddane operatorom genetycznym i tworzą nową populację. W szczególności stosowana jest strategia elitarna	W zależności od strategii: $(1+1)$ — wybór lepszego z dwóch osobników, $(\mu + \lambda)$ — wybór μ osobników ze starej i nowej populacji, (μ, λ) — wybór μ z nowej, liczniejszej populacji	Wybieranych jest μ osobników ze starej populacji (odpowiednio $(\mu + \mu)$)
Zastosowanie	Optymalizacja kombinatoryczna	Optymalizacja w przypadku ciągłych zmiennych niezależnych	Większość zadań optymalizacyjnych
Twórcy	Alex Fraser, lata 50. XX w. John Holland, lata 60. i 70. XX w.	Ingo Rechenberg, Hans-Paul Schwefel, lata 60. XX w.	Lawrence Fogel, 1975

Metody grupowania danych

8.1. Wprowadzenie

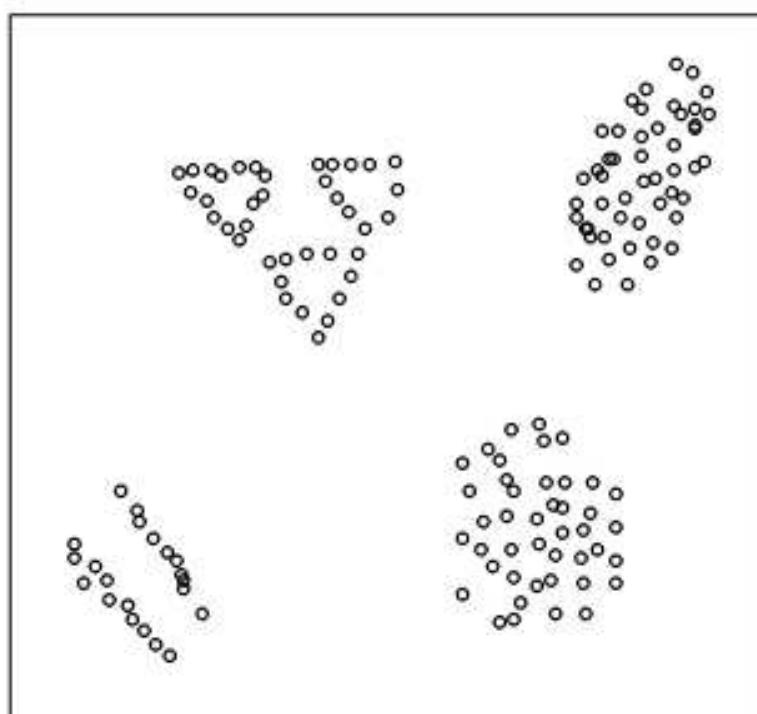
W życiu codziennym i różnych dziedzinach nauki i techniki spotykamy się z wielką, czasami ogromną ilością informacji. Ludziom wystarczy jedno spojrzenie, aby wydobyć z określonego obrazu kształty interesujących nas obiektów. Inteligentne maszyny ciągle jednak są niezdolne do szybkiego i niezawodnego rozpoznawania obiektów w obrazie, ze względu na brak uniwersalnych algorytmów sprawdzających się w każdej sytuacji. Celem grupowania danych jest podział ich zbioru na grupy podobnych danych. Obiektami w zbiorze danych mogą być np. klienci banku, postaci lub przedmioty na zdjęciu, osoby chore i zdrowe. Człowiek może efektywnie pogrupować dane jedynie jedno- i dwuwymiarowe, ale już dane trójwymiarowe mogą przysparzać poważnych trudności. Skalę problemu potęguje fakt, że liczba próbek w rzeczywistych zadaniach może sięgać tysięcy i milionów. W świetle tych faktów bardzo przydatne byłyby algorytmy automatycznego grupowania danych. Wynikiem działania tych algorytmów byłaby ustalona struktura podziału danych, tzn. położenie i kształt grup oraz stopnie przynależności każdej próbki do każdej z grup. Grupowanie danych jest skomplikowanym problemem, gdyż struktury ukryte w zbiorze danych mogą mieć dowolne kształty i rozmiary. Ponadto liczba grup jest zazwyczaj nieznana. Niestety, w literaturze do tej pory nie opracowano algorytmu działającego w przypadku dowolnych kształtów grup.

Wybierając odpowiedni algorytm grupowania, należy wykorzystać wiedzę o problemie opisanym zbiorem danych. W ogólnym przypadku podział danych powinien mieć dwie cechy:

- *homogeniczność w grupach*, tzn. dane w obrębie danej grupy powinny być jak najbardziej podobne do siebie;
- *heterogeniczność pomiędzy grupami*, tzn. dane należące do różnych grup powinny być jak najbardziej różne od siebie.

Podobieństwo wektorów danych może być zdefiniowane na różne sposoby, zależnie od typu grupowanych danych. Ponieważ najczęściej dane opisują cechy obiektów w postaci numerycznej (liczbowej), najodpowiedniejszą miarą podobieństwa jest zmierzenie odległości pomiędzy obiektami. Możemy posłużyć się np. miarą euklidesową, która jest najczęściej używanym sposobem mierzenia podobieństwa obiektów. Grupy danych mogą być reprezentowane na różne sposoby. Najczęściej grupa jest reprezentowana przez jej centralny punkt w przestrzeni danych. Korzystając z różnych miar podobieństwa, możemy

uzyskać odmienne kształty grup, o środku reprezentowanym przez punkt centralny. Na rysunku 8.1 pokazano przykłady rozmaitych grup danych w przestrzeni dwuwymiarowej. Innym sposobem reprezentacji grupy jest opisanie jej przez zależności logiczne.



Rys. 8.1. Przykłady grup danych w przestrzeni dwuwymiarowej

W zadaniach grupowania danych nie dysponujemy tzw. danymi wzorcowymi, pochodząymi od nauczyciela. Zatem proces grupowania danych możemy utożsamiać z uczeniem nienadzorowanym. W tym rozdziale przedstawimy różne sposoby podziału danych oraz algorytmy automatycznego ich grupowania. Omówimy też kryteria jakości grupowania danych.

8.2. Podziały ostre i rozmyte

Dane podlegające grupowaniu będą reprezentowane przez n -wymiarowe wektory $\mathbf{x}_k = [x_{k1}, \dots, x_{kn}]^T$, $\mathbf{x}_k \in R^n$, $k = 1, \dots, M$, składające się z numerycznych wartości opisujących obiekty. Zbiór M wektorów tworzy macierz \mathbf{X} o wymiarach $n \times M$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{M1} \\ x_{12} & x_{22} & \cdots & x_{M2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{Mn} \end{bmatrix}. \quad (8.1)$$

W przypadku klasyfikacji, kolumny macierzy (8.1) są obiektami, a wiersze cechami (atrybutami). W przypadku diagnostyki medycznej obiekty możemy utożsamiać z pacjentami, a cechy z symptomami chorobowymi albo wynikami badań laboratoryjnych tych pacjentów.

Jeśli grupy są reprezentowane przez ich środki, to celem algorytmu grupowania jest uzyskanie c wektorów $\mathbf{v}_i = [v_{i1}, \dots, v_{in}]$, $i = 1, \dots, c$, będących reprezentantami poszczególnych grup w przestrzeni danych. Należy podkreślić, że z obliczeniowego punktu widzenia niezmiernie trudno byłoby przeanalizować wszystkie możliwe podziały M obiektów na c grup, gdyż ich liczba wynosi [48]

$$\frac{1}{c!} \sum_{i=1}^c \binom{c}{i} (-1)^{(c-i)} i^M. \quad (8.2)$$

Przykład 8.1

Rozważmy problem podziału 100 pacjentów ($M = 100$) na 5 różnych grup, charakteryzujących poszczególne przypadki chorobowe ($c = 5$). Łatwo sprawdzić, że stosując wzór (8.2), otrzymujemy w przybliżeniu $6,57 \cdot 10^{67}$ różnych podziałów. Zatem niezmiernie ważne jest znalezienie metod, które dokonają optymalnego podziału bez konieczności przeglądania wszystkich możliwych rezultatów grupowania.

W zadaniach grupowania danych istotną sprawą jest określenie typu podziału danych. W literaturze rozróżniamy podziały ostre, rozmyte i posybilistyczne, przy czym podziały posybilistyczne traktujemy jako modyfikację podziałów rozmytych.

W ostrym grupowaniu danych obiekt całkowicie należy lub nie należy do danej grupy. Celem grupowania jest podział danych na c grup A_i , tak aby

$$\bigcup_{i=1}^c A_i = \mathbf{X}, \quad (8.3)$$

$$A_i \cap A_j = \emptyset, \quad 1 \leq i \neq j \leq c, \quad (8.4)$$

$$\emptyset \subset A_i \subset \mathbf{X}, \quad 1 \leq i \leq c. \quad (8.5)$$

Założenie (8.3) oznacza, że zbiór wszystkich grup zawiera wszystkie wektory danych, a każdy obiekt należy dokładnie do jednej grupy. Grupy są rozłączne (8.4), a żadna z nich nie jest pusta ani też nie zawiera całego zbioru danych \mathbf{X} (8.5). Aby podzielić dane na c grup, wygodnie jest stosować macierz podziału \mathbf{U} o wymiarach $c \times M$, zawierającą stopnie przynależności μ_{ik} k -tej danej \mathbf{x}_k do i -tej grupy, $k = 1, \dots, M$, $i = 1, \dots, c$.

DEFINICJA 8.1

Niech $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ będzie zbiorem skończonym i c , $2 \leq c < M$, liczbą całkowitą. Przestrzeń ostrego podziału zbioru \mathbf{X} definiujemy następująco:

$$Z_1 = \left\{ \mathbf{U} \in R^{c \times M} \mid \mu_{ik} \in \{0, 1\}, \forall i, k; \sum_{i=1}^c \mu_{ik} = 1, \forall k; 0 < \sum_{k=1}^M \mu_{ik} < M, \forall i \right\}. \quad (8.6)$$

Powyższy podział zakłada, że obiekt należy wyłącznie do jednej grupy i nie istnieją puste grupy ani też grupy zawierające wszystkie obiekty.

Przykład 8.2

Rozważmy dane przedstawione na rysunku 8.2. Dla tych danych ostry podział na trzy

grupy ($c = 3$) może być reprezentowany przez następującą macierz \mathbf{U} :

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \quad (8.7)$$

Zauważmy, że obiekt \mathbf{x}_{10} jest przydzielony do grupy 2, chociaż intuicyjnie nie zaliczyćlibyśmy go do żadnej z grup. Jednak podział ostry wymusza przynależność każdego z obiektów do jednej z grup.

Najczęściej rozpatrywany problem nie pozwala na podział danych tak jednoznaczny, jak w definicji 8.1, gdyż obszary występowania grup mogą zachodzić na siebie. Z pomocą w takim przypadku przychodzą algorytmy powodujące, że obiekty mogą należeć do wielu grup jednocześnie z różnymi stopniami przynależności. Jest to naturalne rozszerzenie podziału ostrego, gdzie tak jak w rzeczywistych problemach nie zawsze możemy zaliczyć dany obiekt jednoznacznie do jednej kategorii. Na przykład granice między samochodami małymi, kompaktowymi i dużymi nie są ścisłe określone. Istnieją dwa rodzaje podziału nieostrego: rozmyty i posybilistyczny. W obu podziałach obiekty mogą należeć do dowolnej liczby grup ze stopniem przynależności będącym liczbą z przedziału $[0, 1]$. W podziale rozmytym istnieje dodatkowo ograniczenie przynależności w ramach jednego obiektu, tak aby suma stopni przynależności tego obiektu do każdej z c grup wynosiła 1. Ograniczenie to jest analogiczne do ograniczenia występującego w probabilitycie, dlatego podział ten nazywa się również *podziałem probabilistycznym*.

DEFINICJA 8.2.

Niech $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ będzie skończonym zbiorem i c , $2 \leq c < M$, liczbą całkowitą. Podział rozmyty zbioru \mathbf{X} definiujemy następująco:

$$Z_2 = \left\{ \mathbf{U} \in R^{c \times M} \mid \mu_{ik} \in [0, 1], \forall i, k; \sum_{i=1}^c \mu_{ik} = 1, \forall k; 0 < \sum_{k=1}^M \mu_{ik} < M, \forall i \right\}. \quad (8.8)$$

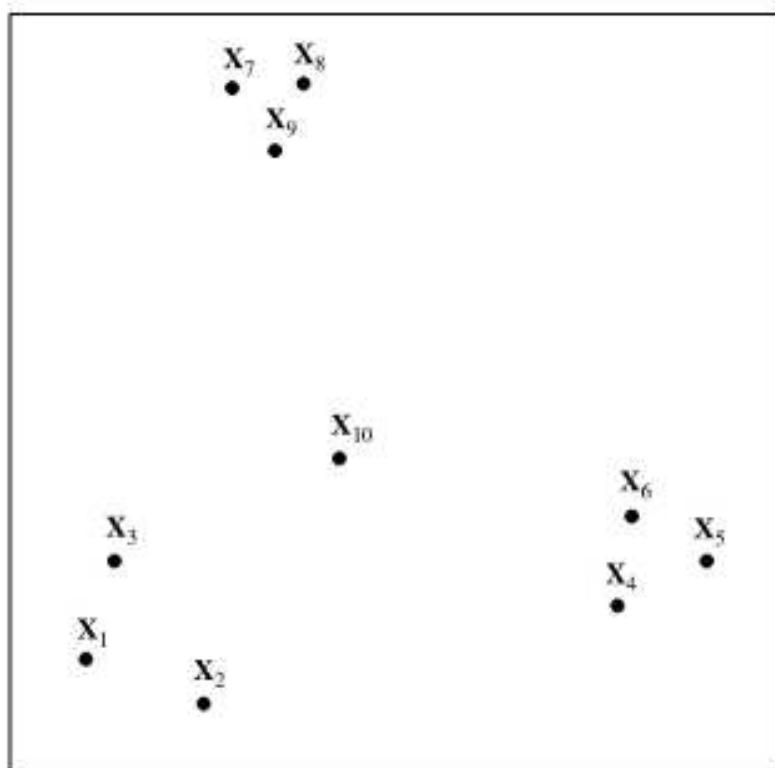
Powyższy podział zakłada, że obiekt może jednocześnie należeć do wszystkich grup z pewnym stopniem przynależności, ale suma wszystkich stopni przynależności musi wynosić 1. Ponadto, nie może być grup pustych, ani też grup zawierających wszystkie dane.

Przykład 8.3

Rozważmy dane przedstawione na rysunku 8.2. Dla tych danych rozmyty podział na trzy grupy może być reprezentowany przez następującą macierz \mathbf{U} :

$$\mathbf{U} = \begin{bmatrix} 0 & 0,06 & 0,02 & 0,98 & 0,98 & 0,99 & 0,01 & 0,01 & 0 & 0,29 \\ 1 & 0,89 & 0,93 & 0,01 & 0,01 & 0,00 & 0,01 & 0,01 & 0 & 0,33 \\ 0 & 0,05 & 0,05 & 0,01 & 0,01 & 0,01 & 0,98 & 0,98 & 1 & 0,38 \end{bmatrix}. \quad (8.9)$$

Zauważmy, że obiekt \mathbf{x}_{10} charakteryzuje się podobnymi stopniami przynależności do wszystkich trzech grup, co odpowiada jego niemal równej odległości od środków tych grup. Obiekt ten możemy utożsamiać z daną przypadkową (szumem). Intuicyjnie przypisalibyśmy szumowi \mathbf{x}_{10} bardzo niskie stopnie przynależności, równe np. 0,1, do wszyst-



Rys. 8.2. Przykładowy zbiór danych

kich trzech grup. Jednak wówczas nie byłby spełniony warunek mówiący, że suma wszystkich stopni przynależności danego obiektu musi wynosić 1.

W literaturze historycznie następnym podziałem był podział posybilistyczny, znośiący ograniczenie sumowania stopni przynależności do jedności. Jedynym ograniczeniem jest, aby obiekt należał przynajmniej do jednej grupy. W praktyce nie jest to duża niedogodność, gdyż mała wartość stopnia przynależności może być traktowana jako brak przynależności.

DEFINICJA 8.3

Niech $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ będzie skończonym zbiorem i c , $2 \leq c < M$, liczbą całkowitą. Podział posybilistyczny zbioru \mathbf{X} jest zdefiniowany następująco:

$$Z_3 = \left\{ \mathbf{U} \in R^{c \times M} \mid \mu_{ik} \in [0, 1], \forall i, k; \forall k, \exists i, \mu_{ik} > 0; 0 < \sum_{k=1}^M \mu_{ik} < M, \forall i \right\}. \quad (8.10)$$

Przykład 8.4

Rozważmy dane przedstawione na rysunku 8.2. Dla tych danych podział posybilistyczny na trzy grupy może być reprezentowany przez następującą macierz \mathbf{U} :

$$\mathbf{U} = \begin{bmatrix} 0,01 & 0,02 & 0,01 & 0,52 & 0,39 & 0,87 & 0,01 & 0,01 & 0,01 & 0,03 \\ 0,87 & 0,44 & 0,79 & 0,04 & 0,03 & 0,03 & 0,05 & 0,04 & 0,05 & 0,12 \\ 0,01 & 0,01 & 0,02 & 0,01 & 0,01 & 0,01 & 0,53 & 0,63 & 0,79 & 0,03 \end{bmatrix}. \quad (8.11)$$

Obecnie nie musi być spełniony warunek sumowania stopni przynależności do jedności. Dlatego szum \mathbf{x}_{10} przynależy do wszystkich grup, ale z niewielkim stopniem przynależności.

8.3. Miary odległości

Ważnym czynnikiem wpływającym na wynik podziału danych jest sposób wyznaczania odległości między obiektami. W przypadku grupowania danych mierzmy odległość w przestrzeni cech, w której istnieją grupowane obiekty i środki (prototypy) grup danych. Najczęściej stosowaną miarą odległości jest *miara euklidesowa*, interpretowana jako geometryczna odległość między dwoma punktami w przestrzeni \mathbf{X} . Rozważmy dwa punkty $\mathbf{x}_d = [x_{d1}, \dots, x_{dn}]^T$ i $\mathbf{v}_i = [v_{i1}, \dots, v_{in}]^T$. *Odległość euklidesową* między tymi punktami definiujemy następująco:

$$D_{id} = \sqrt{\sum_{j=1}^n (x_{dj} - v_{ij})^2} = \|\mathbf{x}_d - \mathbf{v}_i\|_2, \quad (8.12)$$

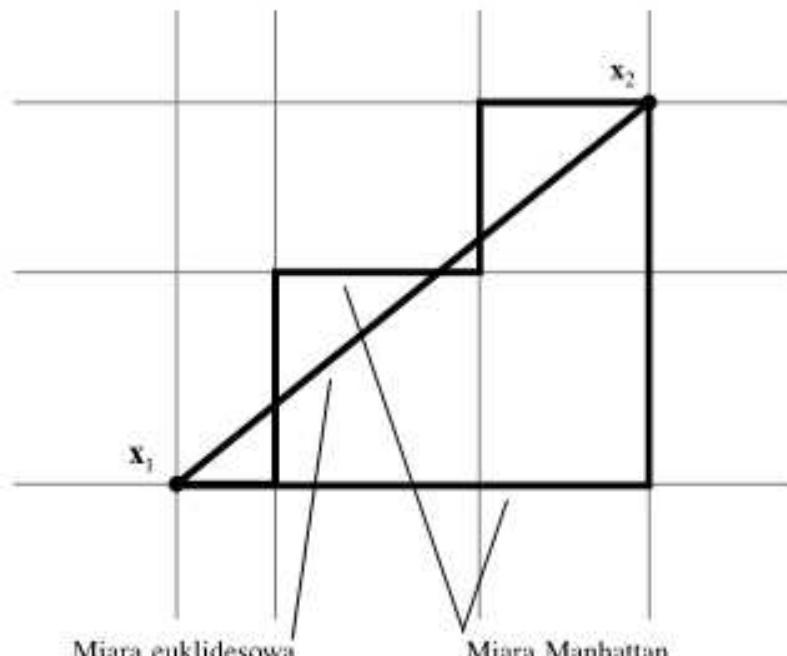
a w zapisie wektorowym

$$D_{id} = [(\mathbf{v}_i - \mathbf{x}_d)^T (\mathbf{v}_i - \mathbf{x}_d)]^{\frac{1}{2}}. \quad (8.13)$$

Miara ta jest uogólnieniem *metryki Minkowskiego*

$$D_{id} = \left(\sum_{j=1}^n |x_{dj} - v_{ij}|^r \right)^{\frac{1}{r}}. \quad (8.14)$$

Dla różnych wartości parametru r możemy uzyskać inne niż euklidesowa miary odległości. Na przykład dla $r = 1$ uzyskujemy *miarę Manhattan* (zwaną też *miarą city block*). Interpretacja tej miary może kojarzyć się z poruszaniem się po ulicach miasta, gdzie jesteśmy zmuszeni trzymać się siatki ulic i dozwolone są tylko zwroty o 90 stopni. Na rysunku 8.3 przedstawiono interpretację miary euklidesowej i Manhattan. W przypadku zmiennych binarnych miara Manhattan nazywana jest *miarą Hamminga*. Miara ta podaje liczbę bitów, o której różnią się dwa ciągi bitów. Ciągi te mogą reprezentować na przykład czarno-białe obrazy.

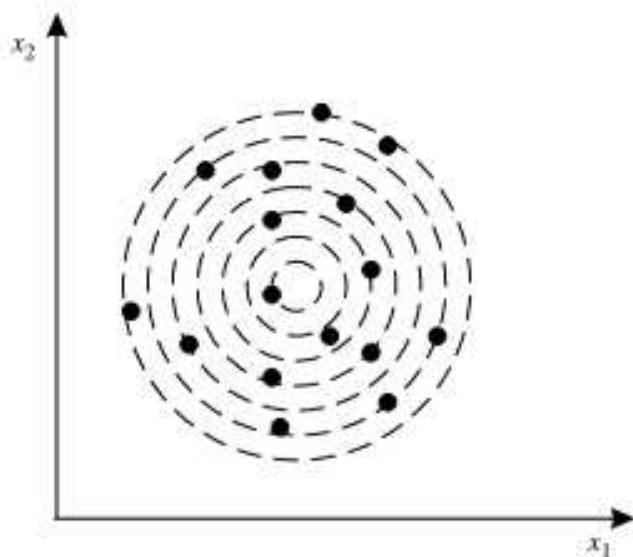


Rys. 8.3. Odległość euklidesowa i Manhattan

Miary Minkowskiego są podatne na różnice w wielkości (skali) poszczególnych zmiennych. Zmienne o dużych wartościach będą dominować nad zmiennymi o małych wartościach, które są na przykład w innej skali. Sposobem na uniknięcie tego problemu może być skalowanie zmiennych, przez co uzyskujemy ważoną normę euklidesową

$$D_{id} = \sqrt{\sum_{j=1}^n w_j (x_{dj} - v_{ij})^2}, \quad (8.15)$$

gdzie w_j jest wagą danego wymiaru. Przyporządkowanie wag poszczególnym zmiennym jest użyteczne, jeśli chcemy uzyskać jednakową ważność zmiennych bez przeskalowania zbioru danych lub narzucić inną hierarchię wymiarów.



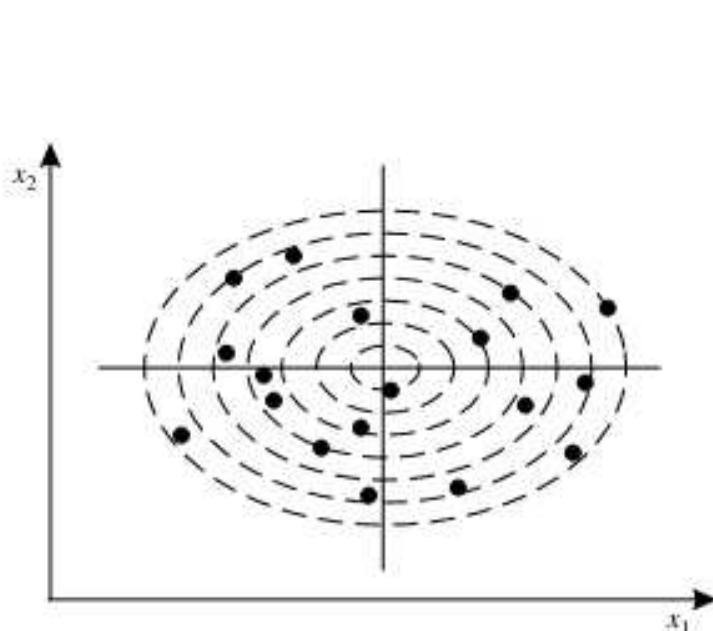
Rys. 8.4. Kształt grupy generowany przez normę euklidesową

Jeśli do miary euklidesowej wprowadzimy dodatkową macierz \mathbf{A} , grupy mogą przybierać kształt elips o dowolnej orientacji. Uzyskujemy wtedy rodzinę norm indukowanych przez iloczyn skalarny. W najprostszym przypadku macierz \mathbf{A} jest macierzą jednostkową, tzn. $\mathbf{A} = \mathbf{I}$. Miara staje się wtedy odległością euklidesową daną wzorem (8.12), a z geometrycznego punktu widzenia grupy tworzą hipersfery. Na rysunku 8.4 pokazano działanie euklidesowej normy dla $n = 2$. Liniami przerywanymi zaznaczono okręgi charakteryzujące się stałą odległością leżących na nich punktów od punktu centralnego (środka). W ogólnym przypadku macierz \mathbf{A} jest macierzą diagonalną o wymiarze $n \times n$, o postaci

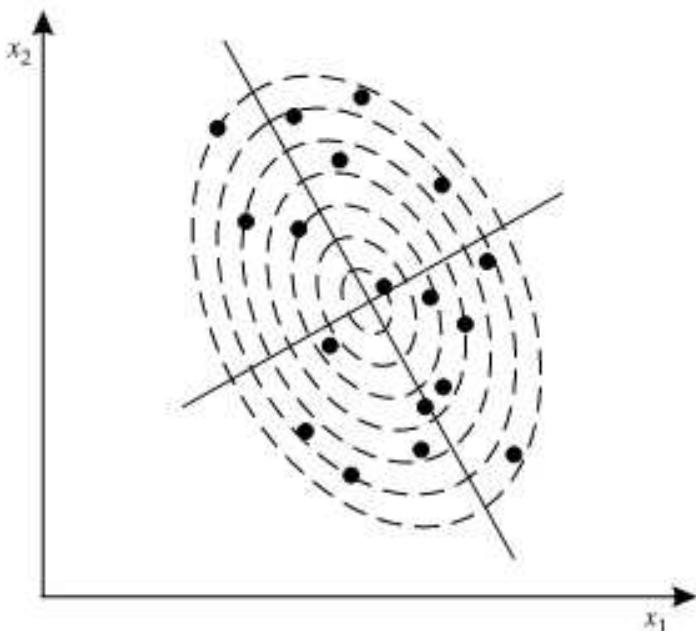
$$\mathbf{A} = \begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_n \end{bmatrix}, \quad (8.16)$$

gdzie $c_i > 0$, $i = 1, \dots, n$. Grupy generowane przez normę z taką macierzą są hiperelipsami z głównymi przekątnymi prostopadłymi do osi przestrzeni danych, co pokazuje rysunek 8.5. Liniami przerywanymi zaznaczono elipsy charakteryzujące się stałą odległością leżących na nich punktów od punktu centralnego.

Pokażemy teraz inny sposób tworzenia macierzy \mathbf{A} . Zdefiniujmy macierz kowariancji danych ze zbioru \mathbf{X}



Rys. 8.5. Kształt grupy generowany przez normę diagonalną



Rys. 8.6. Kształt grupy generowany przez normę Mahalanobisa

$$\mathbf{R} = \frac{1}{M} \sum_{k=1}^M (\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{x}_k - \bar{\mathbf{x}})^T, \quad (8.17)$$

gdzie $\bar{\mathbf{x}}$ oznacza średnią z danych \mathbf{x}_k , $k = 1, \dots, M$. Macierz \mathbf{A} definiujemy następująco:

$$\mathbf{A} = \mathbf{R}^{-1}. \quad (8.18)$$

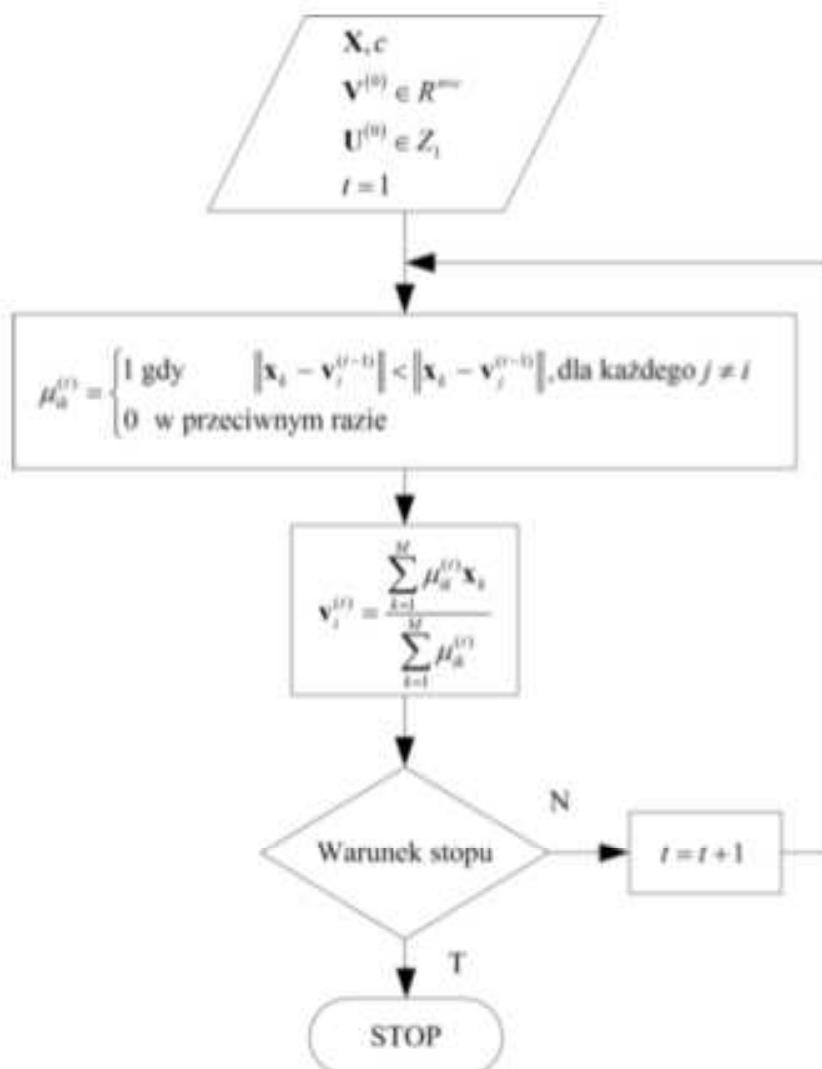
Powstała w ten sposób macierz \mathbf{A} indukuje normę Mahalanobisa w przestrzeni R^n , a grupy są teraz hiperelipsami o dowolnym kształcie i orientacji, co przedstawia rysunek 8.6.

8.4. Algorytm HCM

Algorytm HCM (ang. *Hard C-Means*) dzieli jednoznacznie dane zawarte w macierzy \mathbf{X} na c grup. Realizując ten algorytm, obliczamy odległość między każdym wektorem $\mathbf{x}_k \in R^n$, $k = 1, \dots, M$ i środkiem grupy \mathbf{v}_i , $i = 1, \dots, c$. Środek grupy jest średnią położenia wszystkich obiektów należących do tej grupy. Przynależność do grupy wygodnie jest opisać za pomocą macierzy $\mathbf{U} = [\mu_{ik}] \in Z_1$. Elementami tej macierzy są zera i jedynki mówiące nam o przynależności obiektu \mathbf{x}_k do i -tej grupy. Algorytm przebiega następującymi etapami:

1. Inicjalizacja algorytmu.
2. Wyznaczenie przynależności obiektów do grup na podstawie ich odległości od środków grup.
3. Wyznaczenie nowych środków grup poprzez obliczenie średniej położenia obiektów należących do danej grupy.
4. Sprawdzenie warunku zatrzymania algorytmu. Jeśli warunek nie jest spełniony, następuje przejście do kroku 2.

Inicjalizacja algorytmu polega na wyborze liczby grup c i ustaleniu początkowego położenia ich środków. Położenie to może być wybrane losowo. Alternatywnie początkowe położenie środków może być tożsame z c wektorami \mathbf{x}_k wybranymi losowo lub z c pierwszymi obiektami w zbiorze danych. Szczegółowy przebieg algorytmu jest przedstawiony na rysunku 8.7.



Rys. 8.7. Algorytm HCM

Warunkiem zatrzymania algorytmu najczęściej jest odpowiednio mała zmiana wartości elementów macierzy \mathbf{U} , czyli $\|\mathbf{U}^{(t+1)} - \mathbf{U}^{(t)}\| < \varepsilon$, gdzie ε jest ustaloną stałą. Alternatywnie możemy sprawdzać zmianę położenia środków grup, tzn. $\|\mathbf{V}^{(t+1)} - \mathbf{V}^{(t)}\| < \varepsilon$. Algorytm HCM może dać różne wyniki, zależnie od początkowego położenia środków grup.

8.5. Algorytm FCM

Przedstawimy teraz algorytm FCM (ang. *Fuzzy C-Means*), który pozwala przypisać te same obiekty do różnych grup z odpowiednimi stopniami przynależności. Algorytm FCM jest najczęściej stosowanym algorytmem rozmytego grupowania. Wykrywa on grupy o prototypach będących punktami w przestrzeni danych. Wszystkie grupy mają taki sam kształt zależny od przyjętej z góry normy, gdyż algorytm nie ma możliwości dostosowania macierzy \mathbf{A} do istniejących danych. Algorytm ten wyprowadza się poprzez minimalizację kryterium

$$J(\mathbf{X}; \mathbf{U}, \mathbf{V}) = \sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|_{\mathbf{A}}^2, \quad (8.19)$$

przy czym

$$\mathbf{U} = [\mu_{ik}] \in Z_2 \quad (8.20)$$

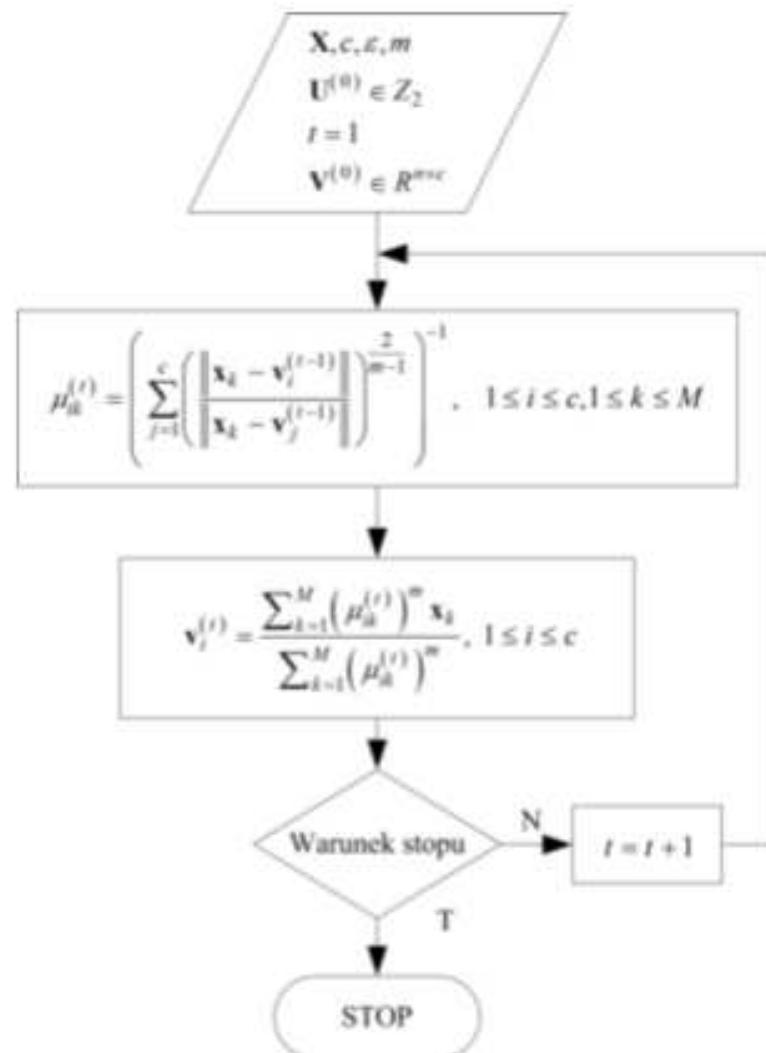
jest macierzą podziału zbioru \mathbf{X} , natomiast

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c] \quad (8.21)$$

jest wektorem średników, które mają być określone w wyniku działania algorytmu, $\mathbf{v}_i \in R^n$, $i = 1, \dots, c$. Występujący we wzorze (8.19) człon

$$D_{ikA}^2 = \|\mathbf{x}_k - \mathbf{v}_i\|_{\mathbf{A}}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A} (\mathbf{x}_k - \mathbf{v}_i) \quad (8.22)$$

pozwala obliczyć odległość między wektorem \mathbf{x}_k i środkiem grupy \mathbf{v}_i , a $m \in (1, \infty)$ jest współczynnikiem mówiącym o stopniu rozmycia powstałych grup danych. Gdy $m \rightarrow 1$, podział staje się coraz bardziej ostry. Gdy $m \rightarrow \infty$, podział staje się coraz bardziej rozmyty (wówczas $\mu_{ik} = 1/c$). W praktyce przyjmuje się wartość $m = 2$. W celu realizacji algorytmu, mając dany zbiór danych \mathbf{X} , należy wybrać liczbę grup c , stopień rozmycia m , parametr ε w kryterium zatrzymania algorytmu oraz zainicjować losowo macierz $\mathbf{U}^{(0)} \in Z_2$ i wektor prototypów grup $\mathbf{V}^{(0)}$. Warunek zatrzymania algorytmu jest taki sam jak w przypadku algorytmu HCM. Algorytm FCM, podobnie jak HCM, może dać różne wyniki zależnie od inicjalizacji. Kształt grup zależy od przyjętej miary odległości. Przebieg działania algorytmu FCM przedstawiony jest na rysunku 8.8.



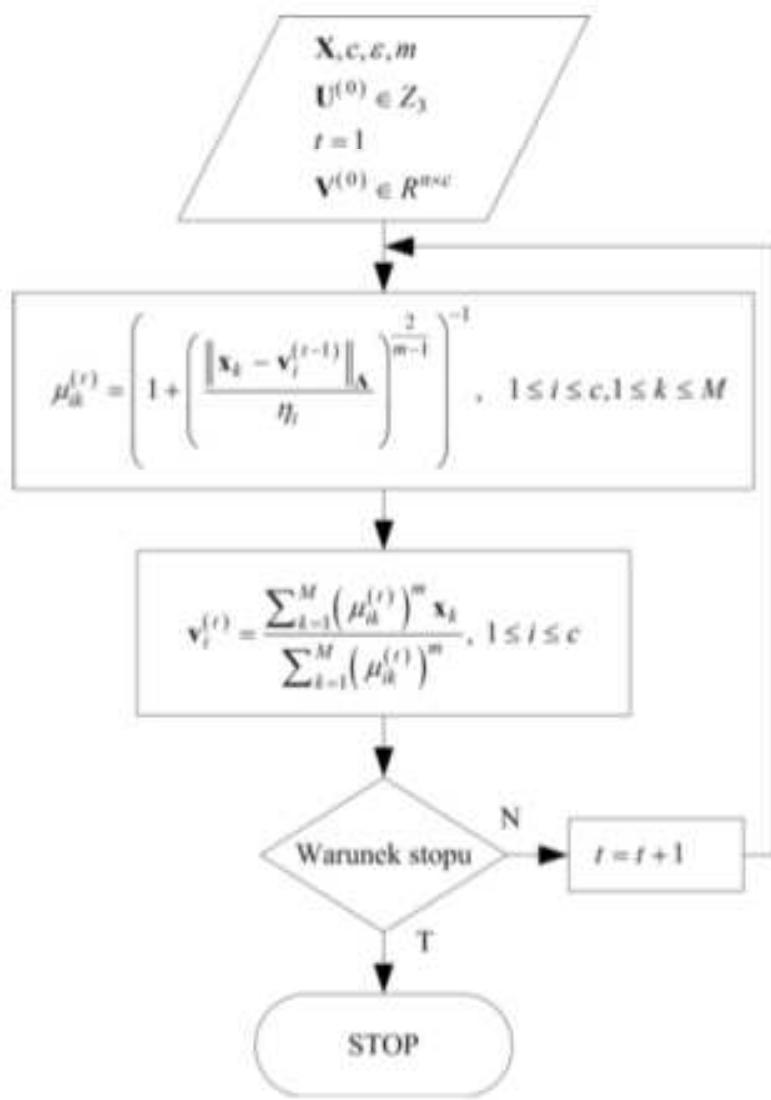
Rys. 8.8. Algorytm FCM

8.6. Algorytm PCM

Wyprowadzając algorytm FCM, zakłada się, że suma stopni przynależności danego obiektu do każdej z grup jest zawsze równa 1. Ograniczenie to może spowodować niepożądane przesunięcie środków grup w sytuacji występowania pojedynczych przypadkowych obiektów (szumu), czasami leżących daleko od właściwych grup danych. Rezygnując z tego ograniczenia, zastosujemy algorytm PCM (ang. *Possibilistic C-Means*), który można otrzymać w wyniku minimalizacji następującej funkcji celu:

$$J(\mathbf{X}, \eta; \mathbf{U}, \mathbf{V}) = \sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|_A^2 + \sum_{i=1}^c \eta_i \sum_{k=1}^M (1 - \mu_{ik})^m, \quad (8.23)$$

w której η_i jest pewną stałą dodatnią. Pierwszy człon kryterium (8.23) jest taki sam jak w kryterium (8.19) dotyczącym algorytmu FCM. Natomiast drugi człon wymusza, aby stopnie przynależności były możliwie duże, bez czego rozwiązanie byłoby osiągnięte dla macierzy \mathbf{U} o elementach równych 0. Takie rozwiązanie byłoby skutkiem rezygnacji z założenia mówiącego o tym, że suma stopni przynależności danego obiektu do każdej z grup jest zawsze równa 1. Łatwo zauważać, że globalną funkcję celu (8.23) można rozłożyć na c funkcji celu dla poszczególnych grup. W wyniku minimalizacji każdej z nich otrzymujemy



Rys. 8.9. Algorytm PCM

$$\mu_{ik} = \left(1 + \left(\frac{D_{ikA}}{\eta_i} \right)^{\frac{2}{m-1}} \right)^{-1} \quad (8.24)$$

gdzie odległość D_{ikA} jest dana wzorem (8.22). Współczynnik η_i określa tzw. szerokość wynikowego rozkładu posybilistycznego. Możemy wybrać tę samą wartość współczynnika η_i dla wszystkich grup lub wyliczyć ją oddziennie dla każdej z grup, proporcjonalnie do średniej odległości obiektów od środka danej grupy, tzn.

$$\eta_i = \frac{\sum_{k=1}^M (\mu_{ik})^m D_{ikA}^2}{\sum_{k=1}^M (\mu_{ik})^m} \quad (8.25)$$

Kryterium zatrzymania algorytmu wybiera się analogicznie jak w przypadku algorytmu HCM. Należy zwrócić uwagę na fakt, że niewłaściwa inicjalizacja algorytmu PCM może prowadzić do podziału, w którym wszystkie stopnie przynależności są równe. Dlatego początkowy podział w algorytmie PCM zazwyczaj następuje z wykorzystaniem algorytmu FCM. Przebieg algorytmu PCM przedstawiony jest na rysunku 8.9.

8.7. Algorytm Gustafsona–Kessela

W prezentowanych do tej pory algorytmach rodzaj normy musi być z góry określony. Dlatego musimy wiedzieć, jakie kształty grup występują w danych. Główną wadą algorytmów ze stałą normą jest szukanie grup o kształcie, który może nie występować w zbiorze danych. Algorytm Gustafsona–Kessela (GK) jest modyfikacją algorytmu FCM. W tym algorytmie z każdą grupą jest skojarzona odrębna macierz \mathbf{A}_i , a odległość między obiektem \mathbf{x}_k a środkiem grupy \mathbf{v}_i wynosi

$$D_{ik}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x}_k - \mathbf{v}_i). \quad (8.26)$$

W trakcie działania algorytmu modyfikowane są także macierze \mathbf{A}_i , $i = 1, \dots, c$, indukujące miarę odległości. Funkcja celu w algorytmie GK jest zdefiniowana tak samo jak w algorytmie FCM (8.19), z tym, że użyta jest miara odległości (8.26). Zatem funkcja celu przybiera postać

$$J(\mathbf{X}; \mathbf{U}, \mathbf{V}, \mathbf{A}) = \sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^m D_{ik}^2. \quad (8.27)$$

gdzie $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_c)$. Zauważmy, że bezpośrednia minimalizacja kryterium (8.27) nie prowadzi do efektywnego rozwiązania, gdyż wartość tego kryterium może być dowolnie mała, np. dla macierzy \mathbf{A}_i z prawie wyłącznie zerowymi elementami. Aby uzyskać poprawny wynik, macierze \mathbf{A}_i muszą być ograniczone, np. poprzez ustalenie wartości ich wyznaczników, tzn.

$$\det(\mathbf{A}_i) = \rho_i, \quad \rho_i > 0, \quad \forall i, \quad i = 1, \dots, c, \quad (8.28)$$

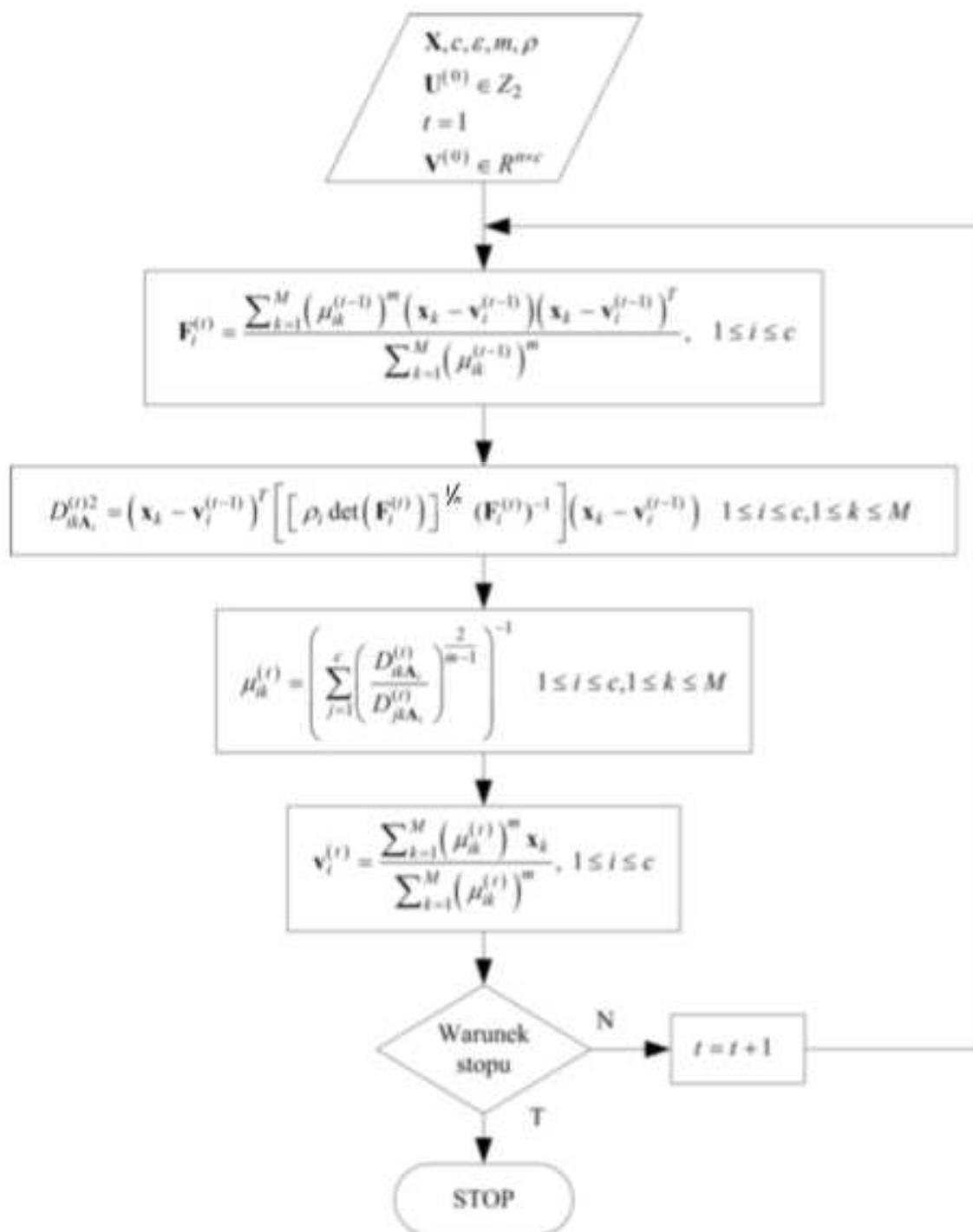
gdzie ρ_i jest przyjętą stałą odzwierciedlającą wiedzę o danych podlegających grupowaniu. W przypadku braku takiej wiedzy przyjmuje się $\rho_i = 1$ dla $i = 1, \dots, c$. Ograniczenie (8.28) powoduje, że objętości grup są stałe, a pozwalały jedynie na zmianę kształtu grup. W wyniku minimalizacji kryterium (8.27) względem macierzy \mathbf{A}_i otrzymujemy

$$\mathbf{A}_i = [\rho_i \det(\mathbf{F}_i)]^{\frac{1}{n}} \mathbf{F}_i^{-1}, \quad (8.29)$$

gdzie \mathbf{F}_i jest tzw. *rozmytą macierzą kowariancji* i -tej grupy

$$\mathbf{F}_i = \frac{\sum_{k=1}^M (\mu_{ik})^m (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^M (\mu_{ik})^m}. \quad (8.30)$$

Inicjalizacja algorytmu wymaga określenia takich samych parametrów jak w algorytmie FCM, a ponadto współczynników ρ_i określających objętości poszczególnych grup (jeśli nie mamy wiedzy o problemie, można przyjąć $\rho_i = 1$). Algorytm GK znajduje grupy dowolnych kształtów, lecz wymaga więcej obliczeń aniżeli algorytm FCM, ze względu na konieczność obliczania wyznacznika i macierzy odwrotnej do macierzy \mathbf{F}_i . Przebieg działania algorytmu GK przedstawiony jest na rysunku 8.10.

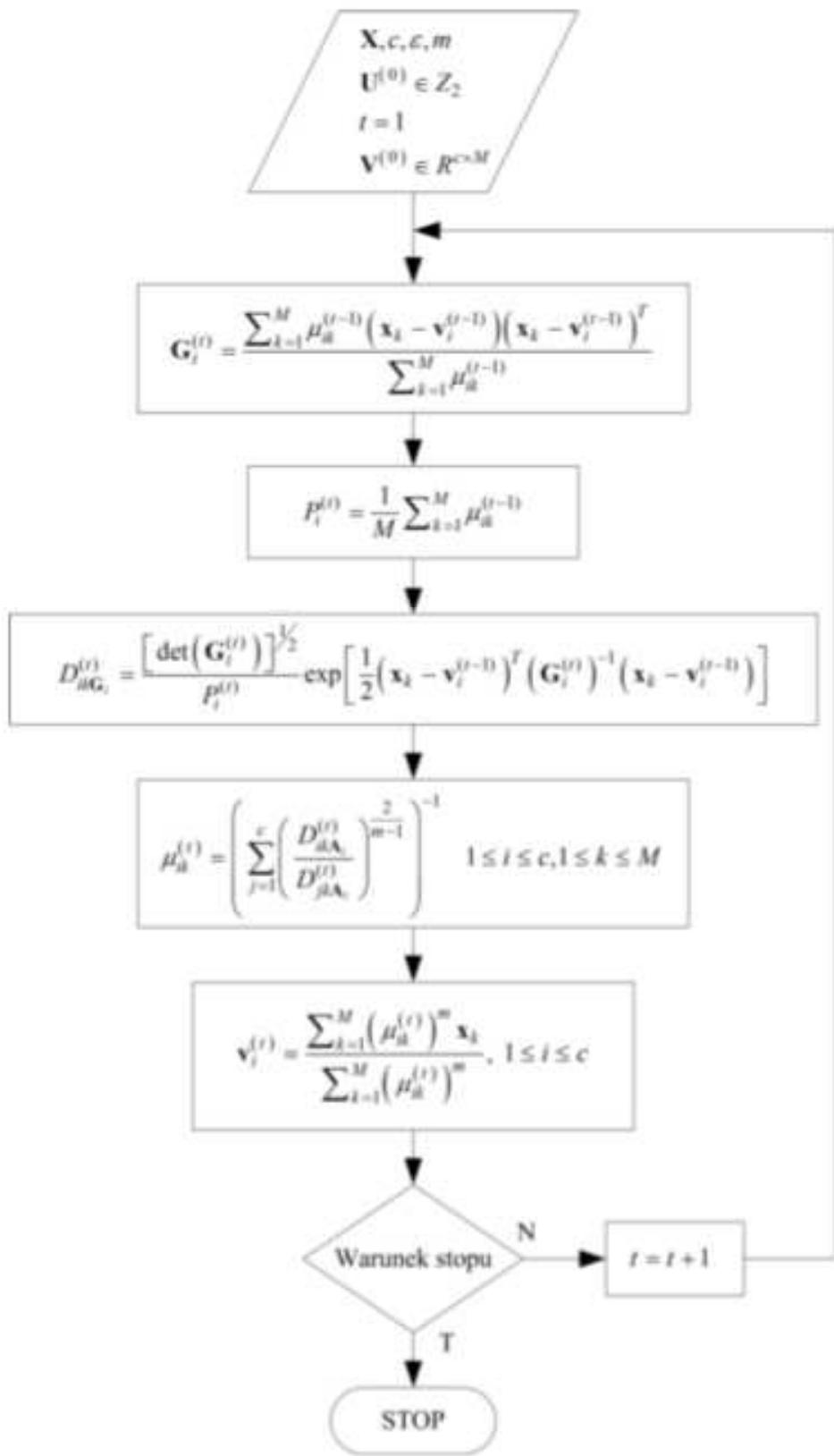


Rys. 8.10. Algorytm Gustafsona-Kessela

8.8. Algorytm FMLE

W algorytmie grupowania FMLE (ang. *Fuzzy Maximum Likelihood Estimates*) miara odległości nawiązuje do postaci estymatorów maksymalnej wiarygodności. Miara ta jest dana wzorem

$$D_{ik} \mathbf{G}_i = \frac{[\det(\mathbf{G}_i)]^{\frac{1}{2}}}{p_i} \exp \left[\frac{1}{2} (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{G}_i^{-1} (\mathbf{x}_k - \mathbf{v}_i) \right], \quad (8.31)$$



Rys. 8.11. Algorytm FMLE

w którym \mathbf{G}_i jest macierzą kowariancji i -tej grupy

$$\mathbf{G}_i = \frac{\sum_{k=1}^M \mu_{ik} (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^M \mu_{ik}}, \quad (8.32)$$

a P_i jest prawdopodobieństwem *a priori* wybrania i -tej grupy

$$P_i = \frac{1}{M} \sum_{k=1}^M \mu_{ik}. \quad (8.33)$$

Stopień przynależności μ_{ik} można interpretować jako prawdopodobieństwo przypisania obiektu \mathbf{x}_k do i -tej grupy. Zbieżność algorytmu zależy silnie od inicjalizacji, gdyż często dąży on do lokalnego minimum. W przeciwieństwie do algorytmu GK, realizacja algorytmu FMLE nie wymaga znajomości lub arbitralnego narzucenia wartości parametru ρ_i dla $i = 1, \dots, c$. Działanie algorytmu przedstawia rysunek 8.11.

8.9. Kryteria jakości grupowania

Liczba grup jest ważnym czynnikiem mającym wpływ na jakość grupowania. Powinna ona odzwierciedlać faktyczną liczbę grup podobnych do siebie obiektów w zbiorze \mathbf{X} . Właściwą liczbę grup można znaleźć poprzez grupowanie zbioru danych dla różnej liczby grup i różnych wartości parametrów (np. parametr m w algorytmie FCM). Każdorazowo też należy dokonać oceny uzyskanego podziału. Ocenę tę przeprowadza się za pomocą specjalnych wskaźników, zwanych *wskaźnikami jakości grupowania*. Poniżej przedstawiamy kilka najbardziej znanych wskaźników jakości grupowania.

a) Wskaźnik odzwierciedlający rozmycie w macierzy podziału \mathbf{U}

Jest to najprostszy wskaźnik mierzący stopień rozmycia macierzy podziału

$$V_1(\mathbf{U}) = \frac{1}{M} \sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^2. \quad (8.34)$$

Najlepszy podział to ten, w którym wskaźnik $V_1(\mathbf{U})$ osiąga wartość maksymalną, czyli

$$\max_c \max_{Z_2} V_1(\mathbf{U}), \quad c = 2, \dots, M-1. \quad (8.35)$$

Współczynnik $V_1(\mathbf{U})$ ocenia odległość wszystkich obiektów od środków grup. Jeżeli każda dana jest silnie związana tylko z jedną grupą, tzn. jeżeli dla każdego k stopień μ_{ik} jest duży tylko dla jednej wartości i , to niepewność danych jest mała, a w konsekwencji $V_1(\mathbf{U})$ przyjmuje dużą wartość. Łatwo zauważać, że wartość wskaźnika (8.34) zależy od odległości poszczególnych obiektów \mathbf{x}_k od środków utworzonych grup. Ze wskaźnikiem (8.34) związany jest wskaźnik określający entropię podziału danych

$$V_2(\mathbf{U}) = -\frac{1}{M} \sum_{i=1}^c \sum_{k=1}^M \mu_{ik} \ln(\mu_{ik}). \quad (8.36)$$

Najlepszym podziałem jest podział, który minimalizuje wskaźnik (8.36), czyli

$$\min_c \left\{ \min_{Z_2} V_2(\mathbf{U}) \right\} \quad c = 2, \dots, M-1. \quad (8.37)$$

Gdy wszystkie stopnie mają wartości bliskie $1/c$, co oznacza wysoki stopień rozmycia grup, wtedy miara $V_2(\mathbf{U})$ przyjmuje duże wartości, co oznacza, że rezultat grupowania jest niezadowalający. Analogicznie, jeśli wszystkie stopnie przynależności μ_{ik} przyjmują wartości bliskie 0 lub 1, to miara $V_2(\mathbf{U})$ przyjmuje małe wartości, co wskazuje na dobry rezultat grupowania.

b) Wskaźnik Fukuyamy–Sugeno

Niedogodnością powyższych wskaźników jest zależność ich wartości od liczby grup c oraz brak powiązania tych wartości z geometrycznym kształtem grup.

Wskaźnik Fukuyamy–Sugeno umożliwia powiązanie podziału z geometrycznymi właściwościami grupowanych danych. Jest on dany następującym wzorem:

$$V_3(\mathbf{U}, \mathbf{V}; \mathbf{X}) = \sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^m \left(\|\mathbf{x}_k - \mathbf{v}_i\|_{\mathbf{A}}^2 - \|\mathbf{x}_k - \bar{\mathbf{v}}\|_{\mathbf{A}}^2 \right), \quad (8.38)$$

w którym $\bar{\mathbf{v}}$ jest średnią wszystkich punktów w zbiorze danych, tzn.

$$\bar{\mathbf{v}} = \frac{1}{M} \sum_{k=1}^M \mathbf{x}_k. \quad (8.39)$$

Optymalny podział danych minimalizuje wskaźnik V_3 .

c) Wskaźnik Xie–Bieni

Wskaźnik Xie–Bieni dany jest wzorem

$$V_4(\mathbf{U}, \mathbf{V}; \mathbf{X}) = \frac{\sum_{i=1}^c \sum_{k=1}^M (\mu_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|^2}{M \left(\min_{i,j} \{\|\mathbf{v}_i - \mathbf{v}_j\|\}^2 \right)}, \quad (8.40)$$

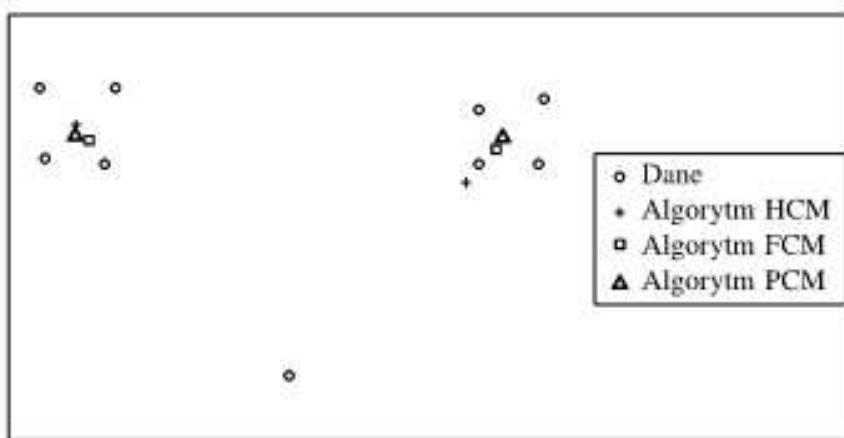
przy czym optymalny dobór liczby klas jest dany wzorem

$$\min_c \left\{ \min_{M_2} V_4(\mathbf{U}) \right\} \quad c = 2, \dots, M-1. \quad (8.41)$$

Najlepszy podział minimalizuje wskaźnik (8.40), będący ilorazem średniej wszystkich odległości pomiędzy grupami i obiektami, i najmniejszej odległości pomiędzy grupami. Właściwa procedura grupowania powinna spowodować, że wszystkie dane będą jak najbliżej środków odpowiednich grup i wszystkie środki będą od siebie jak najbardziej oddalone.

8.10. Ilustracja działania algorytmów grupowania danych

Najczęściej stosowanym algorytmem grupowania danych jest algorytm FCM. Dlatego w tym podrozdziale przeprowadzimy symulację działania tego algorytmu i porównamy z działaniem algorytmów HCM i PCM.



Rys. 8.12. Ilustracja do przykładu 8.5

Przykład 8.5

Na rysunku 8.12 przedstawiono przykładowy zbiór danych składający się z 9 dwuwy- miarowych obiektów, tzn. $M = 9$ oraz $n = 2$. Macierz \mathbf{X} odpowiadająca temu zbiorowi jest postaci

$$\mathbf{X} = \begin{bmatrix} 98 & 97 & 111 & 109 & 178 & 178 & 190 & 189 & 143 \\ 86 & 99 & 99 & 85 & 85 & 95 & 97 & 85 & 46 \end{bmatrix}. \quad (8.42)$$

Latwo zauważać dwie osobne grupy obiektów i jeden obiekt numer 9, „niepasujący” do tych grup. Symbolem „+” oznaczono środki grup uzyskane za pomocą algorytmu HCM. Widać, że obiekt 9 został zakwalifikowany do grupy 2 i wpłynął na położenie środka tej grupy, „ściągając” ten środek w swoim kierunku. W wyniku działania algorytmu HCM powstała następująca macierz podziału:

$$\mathbf{U}_{\text{HCM}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (8.43)$$

Algorytm FCM przydzielił obiekt numer 9 do obu grup z jednakowym stopniem przynależności równym 0,5. W tym przypadku środki obydwu grup są przesunięte w kierunku obiektu numer 9. W wyniku grupowania za pomocą algorytmu FCM powstała następująca macierz podziału:

$$\mathbf{U}_{\text{FCM}} = \begin{bmatrix} 0,99 & 0,98 & 0,98 & 0,99 & 0,00 & 0,01 & 0,02 & 0,01 & 0,50 \\ 0,01 & 0,02 & 0,02 & 0,01 & 1,00 & 0,99 & 0,98 & 0,99 & 0,50 \end{bmatrix}. \quad (8.44)$$

Tabela 8.1. Współrzędne środków grup powsta-
cych w wyniku grupowania zbioru (8.42)

Algorytm HCM	x_1	x_2
Grupa 1	175,60	81,60
Grupa 2	103,75	92,25
Algorytm FCM	x_1	x_2
Grupa 1	106,13	89,38
Grupa 2	181,20	87,73
Algorytm PCM	x_1	x_2
Grupa 1	103,58	90,19
Grupa 2	182,41	89,86

Najlepiej z problemem szumu poradził sobie algorytm PCM, przyporządkowując obiektowi numer 9 stopnie przynależności równe 0,04 i 0,03, co widać, analizując powstałą macierz podziału

$$\mathbf{U}_{PCM} = \begin{bmatrix} 0,76 & 0,56 & 0,53 & 0,73 & 0,03 & 0,03 & 0,02 & 0,02 & 0,04 \\ 0,02 & 0,02 & 0,02 & 0,02 & 0,74 & 0,73 & 0,54 & 0,65 & 0,03 \end{bmatrix}. \quad (8.45)$$

Środki obydwu grup zostały tylko nieznacznie przesunięte w porównaniu z działaniem algorytmów HCM i PCM. W tabeli 8.1 pokazano współrzędne środków grup powstałych w wyniku grupowania zbioru (8.42).

8.11. Uwagi

W tym rozdziale przedstawiliśmy jedynie podstawowe metody grupowania danych. W celu ilustracji ich działania porównaliśmy algorytmy HCM, FCM i PCM. Najczęściej stosowana metoda FCM jest wrażliwa na występowanie zaszumionych danych. Metoda ta może służyć do inicjalizacji odpornej na zakłócenia metody PCM. Stosuje się ją również do wstępnego rozmieszczenia funkcji przynależności podczas projektowania systemów neuronowo-rozmytych (rozdz. 9 i 10). Metody grupowania danych stanowią niezwykle ważne narzędzie badawcze w ramach inteligencji komputerowej i mają liczne zastosowania. Zarówno podstawowe algorytmy przedstawione w tym rozdziale, jak i bardziej zaawansowane metody, np. zorientowane na wykrywanie grup o określonych kształtach, były szczegółowo omawiane w literaturze [3, 9, 23, 34, 83]. Warto wspomnieć, że pionierami badań w zakresie metod grupowania danych są James C. Bezdek i Enrique H. Ruspini, których oryginalne prace przedrukowano w książce [10].

Systemy neuronowo-rozmyte typu Mamdaniego, logicznego i Takagi–Sugeno

9.1. Wprowadzenie

W ciągu ostatnich kilkunastu lat przedstawiono rozmaite struktury sieci neuronowo-rozmytych, często nazywanych w literaturze światowej systemami neuronowo-rozmytymi. Łączą one zalety sieci neuronowych oraz klasycznych systemów rozmytych. W szczególności sieci neuronowo-rozmyte charakteryzują się — w przeciwieństwie do sieci neuronowych — czytelną reprezentacją wiedzy reprezentowanej przez reguły rozmyte. Jak wiadomo, wiedza w sieciach neuronowych jest reprezentowana przez wartości wag synaptycznych, a zatem jest zupełnie nieczytelna na przykład dla użytkownika medycznego systemu ekspertowego, wykorzystującego sieci neuronowe. Ponadto, sieci neuronowo-rozmyte można uczyć, wykorzystując ideę metody wstępnej propagacji błędów, która jest podstawą uczenia wielowarstwowych sieci neuronowych. Uczeniu najczęściej podlegają parametry funkcji przynależności części JEŻELI oraz TO reguł rozmytych. Jak pokazano w rozdziale 7 istnieje również możliwość zastosowania algorytmów ewolucyjnych do uczenia nie tylko parametrów funkcji przynależności, ale również samych reguł rozmytych. Wyżej omówione zalety sieci neuronowo-rozmytych są powodem powszechnego ich stosowania w zadaniach klasyfikacji, aproksymacji i predykcji. W literaturze światowej większość struktur neuronowo-rozmytych wykorzystuje wnioskowanie typu Mamdaniego lub schemat Takagi–Sugeno. Jak wspomnialiśmy w rozdziale 4, wnioskowanie typu Mamdaniego polega na połączeniu poprzedników i następników reguł za pomocą t -normy (najczęściej t -norma typu min lub typu iloczyn). Wówczas agregacja poszczególnych reguł odbywa się za pomocą t -konormy. W przypadku schematu Takagi–Sugeno następni reguł nie mają charakteru rozmytego, ale są funkcjami zmiennych wejściowych. Znacznie rzadziej wykorzystuje się wnioskowanie typu logicznego, które polega na połączeniu poprzedników i następników reguł za pomocą implikacji rozmytej, spełniającej warunki definicji 4.47. W przypadku wnioskowania typu logicznego agregacja poszczególnych reguł odbywa się za pomocą t -normy. Jest oczywiste, że projektanci i użytkownicy systemów neuronowo-rozmytych chcieliby uzyskać dość dużą dokładność działania tych systemów w sensie przyjętego kryterium jakości. W zadaniach aproksymacji i predykcji takim kryterium jakości jest błąd średni kwadratowy, natomiast w zadaniach klasyfikacji jest liczba błędnie sklasyfikowanych próbek. W obu zadaniach eksperymenty przeprowadza się na ciągach uczących oraz ciągach testowych. Należy podkreślić, że zadowalające rezultaty uzyskane na ciągu uczącym nie gwarantują poprawnej pracy systemu na ciągu testowym. Innymi słowy, system neuronowo-rozmyty powinien mieć

dobre właściwości tzw. generalizacji. W szczególności systemy neuronowo-rozmyte zaprojektowane z wykorzystaniem zarówno funkcji przynależności, jak i wag opisujących ważność reguł oraz ważność zmiennych lingwistycznych w poszczególnych regułach powinny charakteryzować się odpowiednią liczbą wszystkich parametrów, które mają być przedmiotem uczenia. Duża liczba parametrów zapewnia mały błąd uczenia, ale zazwyczaj prowadzi do zlej generalizacji. Natomiast mała liczba parametrów w systemie prowadzi do większego błędu uczenia. W tym rozdziale przedstawimy systemy typu Mamdaniego, logicznego i Takagi–Sugeno, algorytmy ich uczenia oraz dokonamy analizy porównawczej efektywności ich działania. Rozwiążemy zagadnienie zaprojektowania systemów neuronowo-rozmytych, które charakteryzują się osiągnięciem kompromisu między błędem działania systemu a liczbą parametrów opisujących ten system.

9.2. Opis wykorzystanych problemów symulacyjnych

Systemy neuronowo-rozmyte rozważane w tym oraz w następnym rozdziale będą testowane z wykorzystaniem standardowych problemów testowych (ang. *benchmarks*).

W tabeli 9.1 podano nazwę problemu, liczbę danych wejściowych, długość ciągu uczącego oraz długość ciągu testowego.

Tabela 9.1. Problemy symulacyjne

Lp.	Nazwa problemu	Liczba wejść	Długość ciągu uczącego	Długość ciągu testowego
1	Polimeryzacja	3	70	20
2	HANG (modelowanie statycznej funkcji nieliniowej)	2	50	20
3	NDP (modelowanie dynamicznej funkcji nieliniowej)	2	1000	200
4	Modelowanie smaku ryżu	5	75	30
5	Rozpoznawanie gatunku wina	13	125	53
6	Klasyfikacja kwiata irysa	4	90	60

Poniżej przedstawiamy szczegółowy opis problemów zestawionych w tabeli 9.1. Informacje o liczbie reguł oraz liczbie epok dotyczą symulacji przeprowadzonych w tym rozdziale (problemy 9.2.1–9.2.4).

9.2.1. Polimeryzacja

Rozważamy problem modelowania procesu wytwarzania polimerów. Urządzenie wytwarza polimery (związki wielkocząsteczkowe otrzymywane z monomerów, czyli związków

małocząsteczkowych) w wyniku reakcji chemicznej zwanej polimeryzacją, podczas której wiele małych cząsteczek tego samego związku łączy się samorzutnie (lub pod wpływem katalizatorów). W celu modelowania systemu wybierane są trzy ciągłe zmienne wejściowe. Należą do nich: stężenie monomeru, zmiana stężenia monomeru oraz jego aktualne tempo przepływu. Na podstawie wartości zmiennych wejściowych należy wyznaczyć następną wartość tempa przepływu monomeru. Badaniom symulacyjnym poddano systemy składające się z 3 wejść, jednego wyjścia oraz 6 reguł. Eksperyment został powtórzony wielokrotnie dla 6000 epok (420 000 iteracji), a jego rezultaty zostały uśrednione.

9.2.2. Modelowanie statycznej funkcji nieliniowej

Jest to zagadnienie aproksymacji funkcji nieliniowej (zwanej w literaturze angielskojęzycznej HANG) opisanej wzorem

$$y(x_1, x_2) = (1 + x_1^{-2} + x_2^{-1.5})^2, \quad (9.1)$$

przy czym $x_1, x_2 \in [1, 5]$. Ciąg uczący składa się z 50 wektorów danych wejściowych i odpowiadających im wartości funkcji. Badaniom symulacyjnym poddano systemy składające się z 2 wejść, jednego wyjścia oraz 8 reguł. Eksperyment został powtórzony wielokrotnie dla 8000 epok (400 000 iteracji), a jego rezultaty zostały uśrednione.

9.2.3. Modelowanie nieliniowego obiektu dynamicznego

Jest to problem modelowania nieliniowego obiektu dynamicznego (NDP, ang. *Nonlinear Dynamic Problem*), którego zachowanie opisane jest wzorem

$$y(t) = g(y(t-1), y(t-2)) + u(t), \quad (9.2)$$

w którym

$$g(u(t-1), y(t-2)) = \frac{y(t-1)y(t-2)(y(t-1) - 0.5)}{1 + y^2(t-1) + y^2(t-2)}, \quad (9.3)$$

a $u(t)$ jest sygnałem wejściowym.

Do celów uczenia systemów neuronowo-rozmytych wykorzystuje się ciąg przykładowych stanów obiektu dla losowego sygnału wejściowego o rozkładzie jednostajnym (pierwsze 500 próbek) oraz dla sinusoidalnego sygnału wejściowego postaci $u(t) = \sin(2\pi t/25)$ (kolejne 500 próbek). Ciąg generowano dla zerowego stanu początkowego. Badaniom symulacyjnym poddano systemy składające się z 3 wejść, jednego wyjścia oraz 6 reguł. Eksperyment został powtórzony wielokrotnie dla 500 epok (500 000 iteracji), a jego rezultaty zostały uśrednione.

9.2.4. Modelowanie smaku ryżu

Problemem do rozwiązania w tym przykładzie jest znalezienie nieliniowej zależności pomiędzy danymi wejściowymi, charakteryzującymi próbki ryżu, a sygnałem wyjściowym,

mającym interpretację smaku ryżu. Dane składają się ze 105 przypadków. Każda próbka została opisana przez 5 cech: aromat, wygląd zewnętrzny, smak, lepkość i twardość, stanowiących dane wejściowe systemu. Wyjściem systemu jest ogólna ocena smaku ryżu. Dane wejściowe i wyjściowe zostały znormalizowane do przedziału [0, 1]. Badaniom symulacyjnym poddano systemy składające się z 2 wejść, jednego wyjścia oraz 6 reguł. Eksperyment został powtórzony wielokrotnie dla 5000 epok (375 000 iteracji), a jego rezultaty zostały uśrednione.

9.2.5. Rozpoznawanie gatunku wina

Zadaniem do rozwiązania jest poprawna klasyfikacja próbek wina. Dane w problemie rozpoznawania wina składają się z chemicznej analizy 178 win z tego samego regionu Włoch, ale dostarczanych z trzech różnych winnic. Dane wejściowe stanowi 13 ciągły atrybutów, do których zaliczymy m.in.: zawartość alkoholu, zawartość kwasu jabłkowego, osad, zasadowość osadu, zawartość magnezu, całkowita zawartość fenolu, intensywność koloru oraz odcień. W omawianym eksperymencie wszystkie dane podzielone zostały na ciąg uczący (125 próbek) oraz ciąg testujący (53 próbki).

9.2.6. Klasyfikacja kwiatu irysa

Problem polega na klasyfikacji kwiatu irysa na podstawie długości liścia w cm, szerokości liścia w cm, długości płatka w cm, szerokości płatka w cm. Rozróżniamy trzy klasy: *Irys Setowa*, *Irys Versicolour* oraz *Irys Virginica*. Dane obejmują 150 zestawów, które podzielono losowo na ciąg uczący (90 zestawów) i testowy (60 zestawów).

Uwaga 9.1

Do uczenia wszystkich systemów neuronowo-rozmytych przedstawionych w tym rozdziale zastosowano algorytmy gradientowe typu momentum ze współczynnikiem uczenia $\eta = 0,25$ oraz ze współczynnikiem momentum 0,1. Algorytmy te wyprowadzono w podrozdziale 9.6 bez uwzględnienia składnika momentum w poszczególnych iteracyjnych procedurach. We wszystkich systemach neuronowo-rozmytych rozważanych w tym rozdziale przyjęto następującą zasadę:

- poszczególne reguły są agregowane za pomocą t -konormy typu max w przypadku systemu Mamdaniego oraz t -normy typu min w przypadku systemu logicznego,
- poprzedniki reguł są agregowane za pomocą t -normy typu iloczyn.

Podstawą oceny systemów neuronowo-rozmytych będzie wartość błędu (błąd średni kwadratowy w przypadku zagadnień aproksymacji lub liczba błędnie sklasyfikowanych próbek w przypadku zagadnień klasyfikacji). Najpierw wyznacza się średni błąd w ramach poszczególnych epok, a następnie wśród tych błędów znajduje się błąd minimalny.

9.3. Systemy neuronowo-rozmyte typu Mamdaniego

Rozważymy dwa rodzaje systemów neuronowo-rozmytych typu Mamdaniego, tzw. systemy typu A i typu B. W obu przypadkach poprzedniki i następni reguły są połączone za pomocą t -normy. W systemach typu A na wyjściu bloku wnioskowania mamy N zbiorów rozmytych, natomiast w systemach typu B na wyjściu tego bloku otrzymujemy jeden zbiór rozmyty, który jest wynikiem agregacji rezultatów wnioskowania w poszczególnych regułach.

9.3.1. Systemy typu A

W systemach typu A wyostrzanie realizowane jest za pomocą zależności

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot \mu_{\bar{B}^r}(\bar{y}^r)}{\sum_{r=1}^N \mu_{\bar{B}^r}(\bar{y}^r)}. \quad (9.4)$$

Funkcje przynależności zbiorów rozmytych \bar{B}^r , $r = 1, 2, \dots, N$, zdefiniowane są za pomocą następującego wzoru:

$$\mu_{\bar{B}^r}(y) = \sup_{\mathbf{x} \in \mathbf{X}} \left\{ \mu_{A^r}(\mathbf{x}) * \mu_{A^r \rightarrow B^r}(\mathbf{x}, y) \right\}. \quad (9.5)$$

Przy rozmywaniu typu singleton wzór (9.5) przybiera postać

$$\mu_{\bar{B}^r}(y) = \mu_{A^r \rightarrow B^r}(\bar{\mathbf{x}}, y) = T(\mu_{A^r}(\bar{\mathbf{x}}), \mu_{B^r}(y)). \quad (9.6)$$

Ponieważ

$$\mu_{A^r}(\bar{\mathbf{x}}) = \frac{n}{i=1} \left(\mu_{A_i^r}(\bar{x}_i) \right), \quad (9.7)$$

więc

$$\mu_{\bar{B}^r}(y) = \mu_{A^r \rightarrow B^r}(\bar{\mathbf{x}}, y) = T \left[\frac{n}{i=1} \left(\mu_{A_i^r}(\bar{x}_i) \right), \mu_{B^r}(y) \right], \quad (9.8)$$

gdzie T jest dowolną t -normą. Korzystając z faktu, że

$$\mu_{B^r}(\bar{y}^r) = 1 \quad (9.9)$$

oraz

$$T(a, 1) = a, \quad (9.10)$$

otrzymuje się następującą zależność:

$$\mu_{\bar{B}^r}(\bar{y}^r) = \frac{n}{i=1} \left(\mu_{A_i^r}(\bar{x}_i) \right). \quad (9.11)$$

Podstawiając zależność (9.11) do wzoru (9.4), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot T_{i=1}^n \left(\mu_{A_i^r}(\bar{x}_i) \right)}{\sum_{r=1}^N T_{i=1}^n \left(\mu_{A_i^r}(\bar{x}_i) \right)}, \quad (9.12)$$

W systemach typu A realizuje się osobne wnioskowanie w obrębie każdej reguły i oblicza się $\mu_{B^r}(\bar{y}^r)$, $r = 1, 2, \dots, N$. Założymy, że zmienne lingwistyczne wejściowe i wyjściowe są opisane gaussowskimi funkcjami przynależności, to znaczy

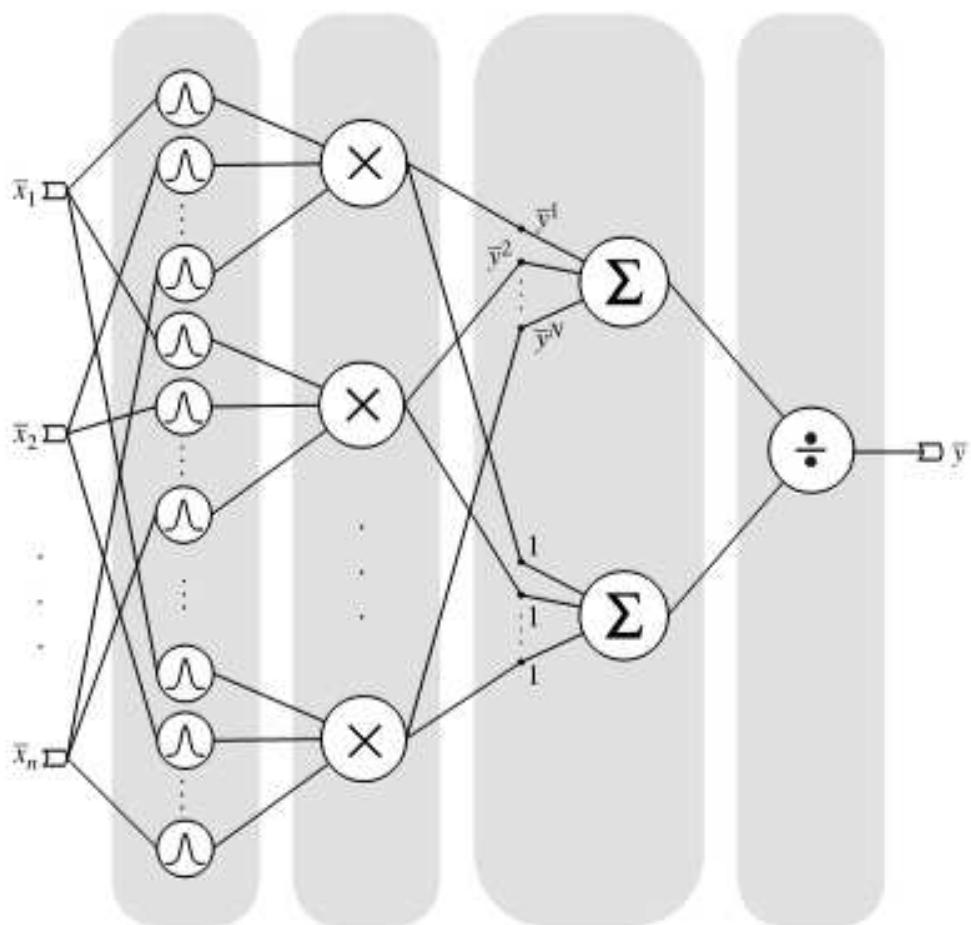
$$\mu_{A'_i}(x_i) = \exp \left[-\left(\frac{x_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], \quad (9.13)$$

$$\mu_{B^r}(y) = \exp \left[-\left(\frac{y - \bar{y}^r}{\sigma^r} \right)^2 \right]. \quad (9.14)$$

Podstawiając powyższe zależności do wzoru (9.4) oraz stosując regułę Larsena, otrzymamy następujący wzór:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \left(\prod_{i=1}^n \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right)}{\sum_{r=1}^N \left(\prod_{i=1}^n \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right)}. \quad (9.15)$$

Zauważmy, że w zależności (9.15) nie występuje parametr σ^r wyjściowego zbioru rozmytego B^r , $r = 1, 2, \dots, N$. Na rysunku 9.1 przedstawiono schemat blokowy struktury odzwierciedlającej zależność (9.15). Jak widać, struktura ta ma charakter wielowarstwowej sieci. Strukturę tę nazywamy siecią neuronowo-rozmytą. Do jej uczenia można zastosować ideę metod wstecznej propagacji błędów.



Rys. 9.1. Sieciowa struktura systemu opisanego wzorem (9.15)

9.3.2. Systemy typu B

W systemach typu B wyostrzanie realizowane jest za pomocą zależności

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \mu_{B^r}(\bar{y}^r)}{\sum_{r=1}^N \mu_{B^r}(\bar{y}^r)}. \quad (9.16)$$

W systemach tych dokonuje się agregacji poszczególnych zbiorów rozmytych \bar{B}^k danych wzorem (9.6), to znaczy wyznacza się zbiór rozmyty B' poprzez operację sumy zbiorów rozmytych \bar{B}^k

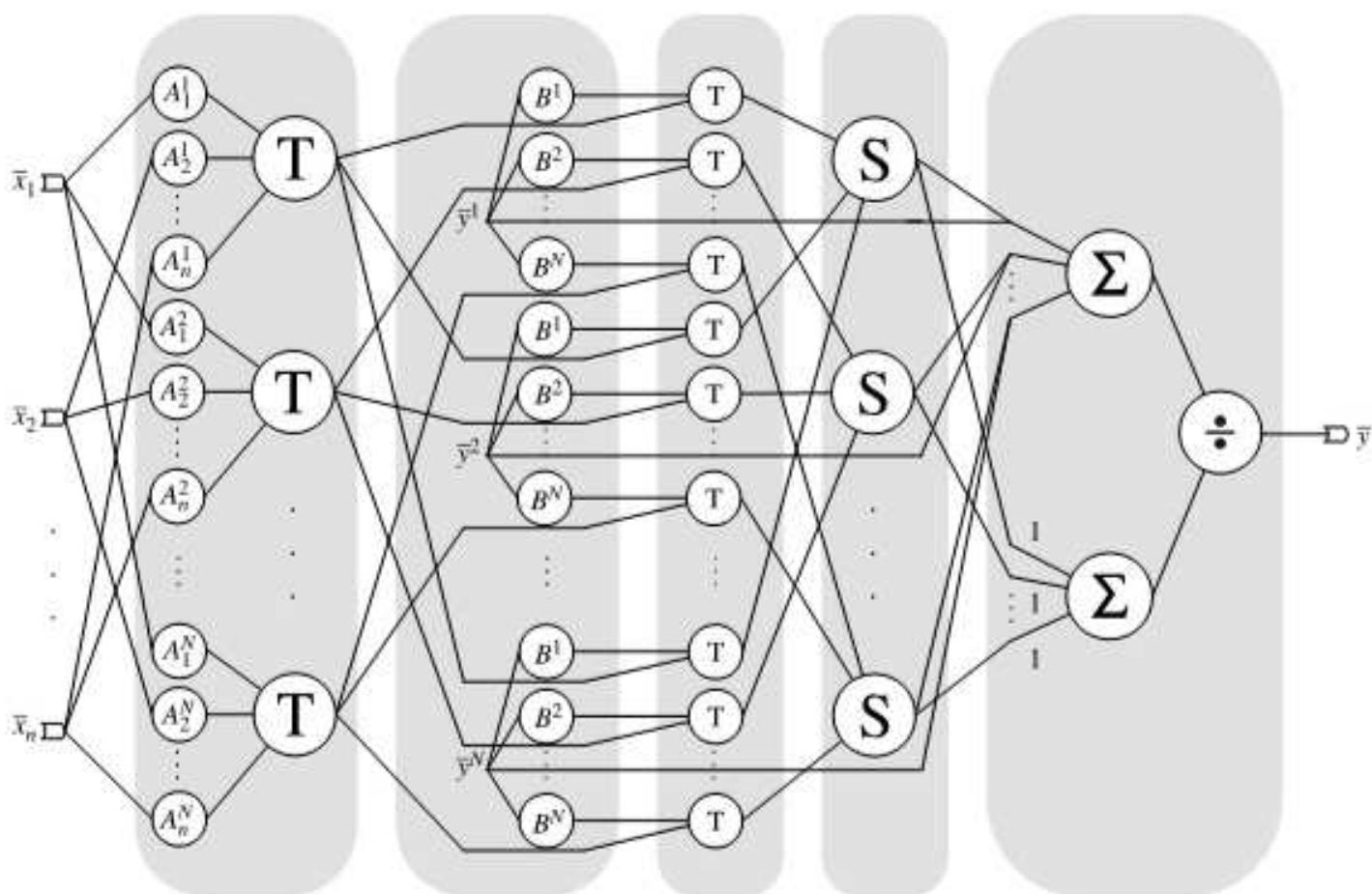
$$B' = \bigcup_{k=1}^N \bar{B}^k. \quad (9.17)$$

Funkcja przynależności zbioru rozmytego B' jest wyznaczona za pomocą t -konormy, to znaczy

$$\mu_{B'}(y) = \sum_{k=1}^N \{\mu_{\bar{B}^k}(y)\}. \quad (9.18)$$

Zatem

$$\begin{aligned} \mu_{B'}(\bar{y}^r) &= \sum_{k=1}^N \{\mu_{\bar{B}^k}(\bar{y}^r)\} = \sum_{k=1}^N \{T(\mu_{A^k}(\bar{x}), \mu_{B^k}(\bar{y}^r))\} \\ &= \sum_{k=1}^N \left\{ T\left(\frac{1}{n} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i), \mu_{B^k}(\bar{y}^r)\right) \right\}. \end{aligned} \quad (9.19)$$



Rys. 9.2. Sieciowa struktura systemu opisanego wzorem (9.20)

Podstawiając wzór (9.19) do zależności (9.16), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ T(T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)) \}}{\sum_{r=1}^N S_{k=1}^N \{ T(T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)) \}}. \quad (9.20)$$

Na rysunku 9.2 pokazano strukturę sieciową systemu opisanego wzorem (9.20).

W systemach typu B również realizuje się osobne wnioskowanie w obrębie każdej reguły, ale następnie dokonuje się agregacji rezultatów wnioskowania w poszczególnych regułach i dopiero wtedy oblicza $\mu_{B^k}(\bar{y}^r)$, $r = 1, 2, \dots, N$.

9.3.3. Systemy typu Mamdaniego w zadaniach modelowania

Systemy typu Mamdaniego zastosujemy do zadań modelowania. Zadania te zostały dokładnie opisane podrozdziale 9.2. Założymy, że zbiory rozmyte A_i^r oraz B^r są charakteryzowane przez gaussowskie funkcje przynależności dane wzorami (9.13) i (9.14).

9.3.3.1. Systemy typu M1

Rozważmy systemy typu Mamdaniego, do których konstrukcji zastosowano definicje norm trójkątnych bez uwzględnienia wag. Korzystając z zależności (9.20) oraz reguły Mamdaniego typu min, otrzymujemy następujący opis systemu neuronowo-rozmytego:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ \min(T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)) \}}{\sum_{r=1}^N S_{k=1}^N \{ \min(T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)) \}}. \quad (9.21)$$

Podstawiając do wzoru (9.21) zależności (9.13) i (9.14) oraz korzystając z treści uwagi 9.1, otrzymujemy

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ \min(T_{i=1}^n \{ \exp[-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \}, \exp[-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]) \}}{\sum_{r=1}^N S_{k=1}^N \{ \min(T_{i=1}^n \{ \exp[-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \}, \exp[-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]) \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot \max_{1 \leq k \leq N} \{ \min(\prod_{i=1}^n \exp[-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \cdot \exp[-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]) \}}{\sum_{r=1}^N \max_{1 \leq k \leq N} \{ \min(\prod_{i=1}^n \exp[-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \cdot \exp[-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]) \}}. \end{aligned} \quad (9.22)$$

Korzystając z zależności (9.20) oraz reguły Mamdaniego typu iloczyn (znanej pod nazwą reguły Larsena), otrzymujemy następujący opis systemu neuronowo-rozmytego:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \} \cdot \mu_{B^k}(\bar{y}^r) \}}{\sum_{r=1}^N S_{k=1}^N \{ T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \} \cdot \mu_{B^k}(\bar{y}^r) \}}. \quad (9.23)$$

Podstawiając do wzoru (9.23) zależności (9.13) i (9.14) oraz korzystając z treści uwagi 9.1, otrzymujemy

$$\begin{aligned}\bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2] \}}{\sum_{r=1}^N S_{k=1}^N \{ T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2] \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot \max_{1 \leq k \leq N} \{ \prod_{i=1}^n \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2] \}}{\sum_{r=1}^N \max_{1 \leq k \leq N} \{ \prod_{i=1}^n \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2] \}}.\end{aligned}\quad (9.24)$$

W dalszym ciągu tego rozdziału już nie będziemy przypominać treści uwagi 9.1. Należy jednak pamiętać, że poszczególne reguły są agregowane za pomocą t -konormy typu max w przypadku systemu Mamdaniego i t -normy typu min w przypadku systemu logicznego oraz że poprzedniki reguł są agregowane za pomocą t -normy typu iloczyn. Systemy neuronowo-rozmyte (9.22) i (9.24) są szczególnymi przypadkami systemu typu B opisanego w punkcie 9.3.2. W systemach (9.22) i (9.24) uczeniu podlegają następujące parametry funkcji przynależności $\bar{x}_i^k, \sigma_i^k, \bar{y}^k, \sigma^k, k = 1, 2, \dots, N$. Przedmiotem badań jest również opisany w punkcie 9.3.1 system Mamdaniego typu A, którego opis dla wygody czytelnika przypominamy poniżej

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot [\prod_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2])] }{\sum_{r=1}^N [\prod_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2])]}. \quad (9.25)$$

System (9.25) został nazwany uproszczoną strukturą Larsena. Przedmiotem uczenia w tym systemie są parametry $\bar{x}_i^r, \sigma_i^r, \bar{y}^r, r = 1, 2, \dots, N$. Można wykazać, że system (9.25) jest szczególnym przypadkiem systemu (9.24). Systemy neuronowo-rozmyte (9.22), (9.24) i (9.25) zastosowano do rozwiązania czterech problemów wymienionych w tabeli 9.1: polimeryzacji, HANG, NDP i modelowania smaku ryżu. Uczeniu metodą wsteczną propagacji błędów poddano wszystkie parametry systemu neuronowo-rozmytego: uczeno środki i szerokości funkcji gaussowskich. W przypadku struktury (9.25) nie występują szerokości nastęników funkcji gaussowskiej.

9.3.3.1.1. Polimeryzacja

W tabeli 9.2 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. W tabeli 9.3 pokazano trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.2. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego	0,0041	3734
Larsena	0,0049	5984
Larsena (uproszczona)	0,0042	4689

Tabela 9.3. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Mamdaniego	1086	1479	1943
Larsena	3621	5984	—
Larsena (uproszczona)	807	2718	4454

Jak wynika z tabeli 9.3, w przypadku struktury Larsena nie udało się nauczyć systemu z błędem 0,0045.

9.3.3.1.2. HANG

W tabeli 9.4 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.5 pokazuje trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.4. Najmniejszy błąd uzyskany w wyniku uczenia

HANG		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego	0,0340	7848
Larsena	0,0387	8000
Larsena (uproszczona)	0,0240	7102

Tabela 9.5. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Mamdaniego	—	—	—
Larsena	—	—	—
Larsena (uproszczona)	4071	6024	7102

Jak wynika z tabeli 9.5, zarówno dla struktury Mamdaniego, jak i Larsena system nie osiągnął żadnego z założonych wartości błędu.

9.3.3.1.3. NDP

W tabeli 9.6 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.6. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego	0,0263	436
Larsena	0,0176	433
Larsena (uproszczona)	0,0140	393

Tabela 9.7. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	NDP		
	0,026	0,023	0,020
Mamdaniego	—	—	—
Larsena	172	233	302
Larsena (uproszczona)	74	82	93

W tabeli 9.7 pokazano trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Jak wynika z tabeli 9.7, struktura Mamdaniego w ogóle nie osiągnęła zadanych wartości błędów.

9.3.3.1.4. Modelowanie smaku ryżu

W tabeli 9.8 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.8. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego	0,0244	4459
Larsena	0,0252	2501
Larsena (uproszczona)	0,0205	3888

Tabela 9.9. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	MODELOWANIE SMAKU RYŻU		
	Wartość błędu		
	0,028	0,025	0,022
Mamdaniego	233	1978	—
Larsena	506	—	—
Larsena (uproszczona)	67	451	2936

Jak wynika z tabeli 9.9, jedynie uproszczona struktura Larsena osiągnęła wszystkie zadane wartości błędów.

9.3.3.2. Systemy typu M2

Rozważmy systemy typu Mamdaniego, do których konstrukcji zastosowano definicje norm trójkątnych z uwzględnieniem wag w_k , charakteryzujących ważność poszczególnych reguł. Korzystając z definicji ważonej t -konormy oraz zależności (9.22), (9.24) i (9.25), otrzymujemy następujące systemy neuronowo-rozmyte:

a) System Mamdaniego z wagami reguł

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ \min (T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ \min (T_{i=1}^n \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r)), w_k \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ \min (T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \}, \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ \min (T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \}, \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]), w_k \}}. \end{aligned} \quad (9.26)$$

b) System Larsena z wagami reguł

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ T_{i=1}^n (\{ \mu_{A_i^k}(\bar{x}_i) \} \cdot \mu_{B^k}(\bar{y}^r)), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ T_{i=1}^n (\{ \mu_{A_i^k}(\bar{x}_i) \} \cdot \mu_{B^k}(\bar{y}^r)), w_k \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2], w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ T_{i=1}^n \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2] \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2], w_k \}}. \end{aligned} \quad (9.27)$$

Przedmiotem uczenia w obu systemach a) i b) są parametry funkcji przynależności \bar{x}_i^k , σ_i^k , \bar{y}^k , σ^k oraz wagi w_k .

c) Uproszczony system Larsena z wagami reguł

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot w_r \cdot \prod_{i=1}^n (\mu_{A_i^r}(\bar{x}_i))}{\sum_{r=1}^N w_r \cdot \prod_{i=1}^n (\mu_{A_i^r}(\bar{x}_i))} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot w_r \cdot \prod_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2])}{\sum_{r=1}^N w_r \cdot \prod_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2])}. \end{aligned} \quad (9.28)$$

Przedmiotem uczenia w systemie c) są parametry funkcji przynależności \bar{x}_i^r , σ_i^r , \bar{y}^r oraz wagi w_r . Systemy neuronowo-rozmyte (9.26), (9.27) i (9.28) zastosowano do rozwiązania czterech problemów wymienionych w tabeli 9.1.

9.3.3.2.1. Polimeryzacja

W tabeli 9.10 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.10. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami	0,0039	4088
Larsena z wagami	0,0043	4501
Larsena (uproszczona) z wagami	0,0039	3691

Tabela 9.11. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Mamdaniego z wagami	26	44	2440
Larsena z wagami	2646	3154	4099
Larsena (uproszczona) z wagami	1633	1633	3443

W tabeli 9.11 pokazano trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

9.3.3.2.2. HANG

W tabeli 9.12 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.13 przedstawia trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.12. Najmniejszy błąd uzyskany w wyniku uczenia

HANG		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami	0,0318	7848
Larsena z wagami	0,0353	6773
Larsena (uproszczona) z wagami	0,0183	1955

Tabela 9.13. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Mamdaniego z wagami	—	—	—
Larsena z wagami	—	—	—
Larsena (uproszczona) z wagami	191	366	632

Jak wynika z tabeli 9.13, zarówno dla struktury Mamdaniego, jak i Larsena nie udało się nauczyć systemu tak, aby uzyskać założony błąd.

9.3.3.2.3. NDP

W tabeli 9.14 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.15 pokazuje trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.14. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami	0,0238	389
Larsena z wagami	0,0164	495
Larsena (uproszczona) z wagami	0,0136	487

Tabela 9.15. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,028	0,020	0,023
Mamdaniego z wagami	157	—	—
Larsena z wagami	121	180	272
Larsena (uproszczona) z wagami	55	59	90

Jak wynika z tabeli 9.15, dla struktury Mamdaniego z wagami reguł nie udało się nauczyć systemu tak, aby uzyskać błąd 0,020 i 0,023.

9.3.3.2.4. Modelowanie smaku ryżu

W tabeli 9.16 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.17 pokazuje trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.16. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami	0,0178	4978
Larsena z wagami	0,0229	4154
Larsena (uproszczona) z wagami	0,0199	4935

Tabela 9.17. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Mamdaniego z wagami	335	716	1751
Larsena z wagami	562	1479	—
Larsena (uproszczona) z wagami	421	852	3264

Jak wynika z tabeli 9.17, dla struktury Larsena z wagami reguł nie udało się nauczyć systemu tak, aby uzyskać błąd 0,022.

9.3.3.3. Systemy typu M3

Rozważymy systemy typu Mamdaniego, do których konstrukcji zastosowano definicje norm trójkątnych z uwzględnieniem wag w_k , charakteryzujących ważność poszczególnych reguł, oraz wag $w_{i,k}$, charakteryzujących ważność poszczególnych zmiennych lingwistycznych wejściowych. Korzystając z definicji ważonej t -konormy oraz zależności (9.22), (9.24) i (9.25), otrzymujemy następujące systemy neuronowo-rozmyte:

a) System Mamdaniego z wagami wejścia i reguł

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ \min (T_{i=1}^{*n} \{ \mu_{A_i^k}(\bar{x}_i), w_{i,k} \}, \mu_{B^k}(\bar{y}^r)), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ \min (T_{i=1}^{*n} \{ \mu_{A_i^k}(\bar{x}_i), w_{i,k} \}, \mu_{B^k}(\bar{y}^r)), w_k \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ \min (T_{i=1}^{*n} \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2], w_{i,k} \}, \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ \min (T_{i=1}^{*n} \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2], w_{i,k} \}, \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2]), w_k \}}. \end{aligned} \quad (9.29)$$

b) System Larsena z wagami wejścia i reguł

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ T_{i=1}^{*n} \{ \mu_{A_i^k}(\bar{x}_i), w_{i,k} \} \cdot \mu_{B^k}(\bar{y}^r), w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ T_{i=1}^{*n} \{ \mu_{A_i^k}(\bar{x}_i), w_{i,k} \} \cdot \mu_{B^k}(\bar{y}^r), w_k \}} \\ &= \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^{*N} \{ T_{i=1}^{*n} \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2], w_{i,k} \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2], w_k \}}{\sum_{r=1}^N S_{k=1}^{*N} \{ T_{i=1}^{*n} \{ \exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2], w_{i,k} \} \cdot \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k})^2], w_k \}}. \end{aligned} \quad (9.30)$$

Przedmiotem uczenia w systemach a) i b) są parametry funkcji przynależności \bar{x}_i^k , σ_i^k , \bar{y}^k , σ^k oraz wag $w_{i,k}$ i w_k .

c) Uproszczony system Larsena z wagami wejścia i reguł

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot w_r [T_{i=1}^n \{ 1 - w_{i,r} (1 - \mu_{A_i^r}(\bar{x}_i)) \}]}{\sum_{r=1}^N w_r [T_{i=1}^n \{ 1 - w_{i,r} (1 - \mu_{A_i^r}(\bar{x}_i)) \}]}$$

$$= \frac{\sum_{r=1}^N \bar{y}^r \cdot w_r [T_{i=1}^n \{1 - w_{i,r} (1 - (\exp[-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2]))\}]}{\sum_{r=1}^N w_r [T_{i=1}^n \{1 - w_{i,r} (1 - (\exp[-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2]))\}]} \quad (9.31)$$

Przedmiotem uczenia w systemie c) są parametry funkcji przynależności \bar{x}_i^r , σ_i^r , \bar{y}^r oraz wagi $w_{i,r}$ i w_r . Systemy neuronowo-rozmyte (9.29), (9.30) i (9.31) zastosowano do rozwiązania czterech problemów wymienionych w tabeli 9.1.

9.3.3.3.1. Polimeryzacja

W tabeli 9.18 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.19 przedstawia trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.18. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami wejść i reguł	0,0034	4704
Larsena z wagami wejść i reguł	0,0035	3822
Larsena (uproszczona) z wagami wejść i reguł	0,0031	2953

Tabela 9.19. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Mamdaniego z wagami wejść i reguł	1915	2303	2549
Larsena z wagami wejść i reguł	1	1	1
Larsena (uproszczona) z wagami wejść i reguł	1	6	13

9.3.3.3.2. HANG

W tabeli 9.20 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.21 przedstawia trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.20. Najmniejszy błąd uzyskany w wyniku uczenia

HANG		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami wejść i reguł	0,0209	5474
Larsena z wagami wejść i reguł	0,0346	1541
Larsena (uproszczona) z wagami wejść i reguł	0,0124	4252

Tabela 9.21. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Mamdaniego z wagami wejść i reguł	4213	5474	5474
Larsena z wagami wejść i reguł	—	—	—
Larsena (uproszczona) z wagami wejść i reguł	628	750	750

Jak wynika z tabeli 9.21, zadanych wartości błędu nie udało się uzyskać dla struktury Larsena z wagami wejść i reguł.

9.3.3.3.3. NDP

W tabeli 9.22 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Tabela 9.23 przedstawia trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.22. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami wejść i reguł	0,0181	498
Larsena z wagami wejść i reguł	0,0146	500
Larsena (uproszczona) z wagami wejść i reguł	0,0188	484

Tabela 9.23. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Mamdaniego z wagami wejść i reguł	60	126	294
Larsena z wagami wejść i reguł	24	31	74
Larsena (uproszczona) z wagami wejść i reguł	—	—	—

Jak wynika z tabeli 9.23, zadanych wartości błędu nie udało się uzyskać dla uproszczonej struktury Larsena z wagami wejść i reguł.

9.3.3.3.4. Modelowanie smaku ryżu

W tabeli 9.24 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.24. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Mamdaniego z wagami wejść i reguł	0,0168	2218
Larsena z wagami wejść i reguł	0,0218	2325
Larsena (uproszczona) z wagami wejść i reguł	0,0190	4975

W tabeli 9.25 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.25. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,028	0,025	0,022
Mamdaniego z wagami wejść i reguł	1	3	3
Larsena z wagami wejść i reguł	295	528	2325
Larsena (uproszczona) z wagami wejść i reguł	1	1	5

9.4. Systemy neuronowo-rozmyte typu logicznego

W poprzednim podrozdziale omówiliśmy systemy neuronowo-rozmyte z wnioskowaniem typu Mamdaniego. Obecnie przedmiotem rozważań będą systemy, w których poprzedniki i następcy reguł są ze sobą powiązane za pomocą implikacji rozmytej.

W systemach typu logicznego wystrzanie realizowane jest za pomocą zależności

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \mu_{B'}(\bar{y}^r)}{\sum_{r=1}^N \mu_{B'}(\bar{y}^r)}. \quad (9.32)$$

W systemach tych zbiór rozmyty B' powstaje w wyniku przecięcia zbiorów rozmytych \bar{B}^k , tzn.

$$B' = \bigcap_{k=1}^N \bar{B}^k. \quad (9.33)$$

Funkcja przynależności zbioru rozmytego B' jest wyznaczona za pomocą t -normy, co zapisujemy następująco:

$$\mu_{B'}(y) = T_{k=1}^N \{\mu_{\bar{B}^k}(y)\}. \quad (9.34)$$

Korzystając ze wzorów (9.34), (9.6) i (9.7), możemy zapisać

$$\begin{aligned}\mu_{B^k}(\bar{y}^r) &= \sum_{k=1}^N \{\mu_{\bar{B}^k}(\bar{y}^r)\} = \sum_{k=1}^N \{I(\mu_{A^k}(\bar{x}), \mu_{B^k}(\bar{y}^r))\} \\ &= \sum_{k=1}^N \left\{ I\left(\frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i), \mu_{B^k}(\bar{y}^r)\right) \right\},\end{aligned}\quad (9.35)$$

gdzie I jest rozmytą implikacją zdefiniowaną w punkcie 4.8.4. Podstawiając wzór (9.35) do zależności (9.32), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot T_{k=1}^N \{I(T_{i=1}^n \{\mu_{A_i^k}(\bar{x}_i)\}, \mu_{B^k}(\bar{y}^r))\}}{\sum_{r=1}^N T_{k=1}^N \{I(T_{i=1}^n \{\mu_{A_i^k}(\bar{x}_i)\}, \mu_{B^k}(\bar{y}^r))\}}. \quad (9.36)$$

Konkretna postać wzoru (9.36) zależy od przyjętej definicji funkcji I . Systemy typu logicznego zastosujemy do rozwiązywania zadań modelowania. Rozważymy systemy M1 (bez wag), systemy M2 (z wagami reguł) i systemy M3 (z wagami reguł i wagami zmiennych lingwistycznych wejściowych). Zastosujemy rozmytą implikację Łukasiewicza, binarną, Reichenbacha, Zadeha oraz Willmotta. Ponadto przedstawimy i przetestujemy uproszczone struktury neuronowo-rozmyte z wykorzystaniem implikacji Łukasiewicza i Zadeha.

9.4.1. Systemy typu M1

Rozważmy systemy typu logicznego, do których konstrukcji zastosowano definicje norm trójkątnych bez uwzględnienia wag. Najpierw wykorzystamy implikację Łukasiewicza. W wyniku zastosowania tej implikacji otrzymujemy następującą zależność:

$$\begin{aligned}\mu_{A^k \rightarrow B^k}(\bar{x}, y) &= I(\mu_{A^k}(\bar{x}), \mu_{B^k}(y)) = I\left(\frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i), \mu_{B^k}(y)\right) \\ &= \min \left[1, 1 - \frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i) + \mu_{B^k}(y) \right].\end{aligned}\quad (9.37)$$

Podstawiając zależność (9.37) do wzoru (9.36), mamy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ \min \left[1, 1 - \frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i) + \mu_{B^k}(\bar{y}^r) \right] \}}{\sum_{r=1}^N T_{k=1}^N \{ \min \left[1, 1 - \frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i) + \mu_{B^k}(\bar{y}^r) \right] \}}. \quad (9.38)$$

Podstawiając do wzoru (9.38) zależności (9.13) i (9.14), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ \min \left[1, 1 - T_{i=1}^n \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}^k}{\sigma_i^k} \right)^2 \right] \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k} \right)^2 \right] \] \}}{\sum_{r=1}^N T_{k=1}^N \{ \min \left[1, 1 - T_{i=1}^n \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}^k}{\sigma_i^k} \right)^2 \right] \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k} \right)^2 \right] \] \}}. \quad (9.39)$$

Stosując rozmytą implikację binarną, otrzymujemy

$$\begin{aligned}\mu_{A^k \rightarrow B^k}(\bar{x}, y) &= I(\mu_{A^k}(\bar{x}), \mu_{B^k}(y)) = I\left(\frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i), \mu_{B^k}(y)\right) \\ &= \max \left[1 - \frac{n}{T} \sum_{i=1}^n \mu_{A_i^k}(\bar{x}_i), \mu_{B^k}(y) \right].\end{aligned}\quad (9.40)$$

Podstawiając zależność (9.40) do wzoru (9.36), mamy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ \max [1 - T_{i=1}^n (\mu_{A_i^k}(\bar{x}_i)), \mu_{B^k}(\bar{y}^r)] \}}{\sum_{r=1}^N T_{k=1}^N \{ \max [1 - T_{i=1}^n (\mu_{A_i^k}(\bar{x}_i)), \mu_{B^k}(\bar{y}^r)] \}}. \quad (9.41)$$

Podstawiając do wzoru (9.41) zależności (9.13) i (9.14), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ \max [1 - T_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2]), \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k})^2]] \}}{\sum_{r=1}^N T_{k=1}^N \{ \max [1 - T_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2]), \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k})^2]] \}}. \quad (9.42)$$

Wynikiem zastosowania rozmytej implikacji Reichenbacha jest następujący wzór:

$$\begin{aligned} \mu_{A^k \rightarrow B^k}(\bar{x}, y) &= I(\mu_{A^k}(\bar{x}), \mu_{B^k}(y)) = I\left(\frac{n}{i=1} (\mu_{A_i^k}(\bar{x}_i)), \mu_{B^k}(y)\right) \\ &= 1 - \frac{n}{i=1} (\mu_{A_i^k}(\bar{x}_i)) (1 - \mu_{B^k}(y)). \end{aligned} \quad (9.43)$$

Podstawiając zależność (9.43) do wzoru (9.36), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ 1 - T_{i=1}^n (\mu_{A_i^k}(\bar{x}_i)) (1 - \mu_{B^k}(\bar{y}^r)) \}}{\sum_{r=1}^N T_{k=1}^N \{ 1 - T_{i=1}^n (\mu_{A_i^k}(\bar{x}_i)) (1 - \mu_{B^k}(\bar{y}^r)) \}}. \quad (9.44)$$

Podstawiając do wzoru (9.44) zależności (9.13) i (9.14), otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^N \{ 1 - T_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2]) (1 - \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k})^2]) \}}{\sum_{r=1}^N T_{k=1}^N \{ 1 - T_{i=1}^n (\exp [-(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k})^2]) (1 - \exp [-(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k})^2]) \}}. \quad (9.45)$$

Wynikiem zastosowania rozmytej implikacji Zadeha jest następujący wzór:

$$\begin{aligned} \mu_{A^k \rightarrow B^k}(\bar{x}, y) &= I(\mu_{A^k}(\bar{x}), \mu_{B^k}(y)) = I\left(\frac{n}{i=1} (\mu_{A_i^k}(\bar{x}_i)), \mu_{B^k}(y)\right) \\ &= \max \left\{ \min \left[\frac{n}{i=1} (\mu_{A_i^k}(\bar{x}_i)), \mu_{B^k}(y) \right], 1 - \frac{n}{i=1} (\mu_{A_i^k}(\bar{x}_i)) \right\}. \end{aligned} \quad (9.46)$$

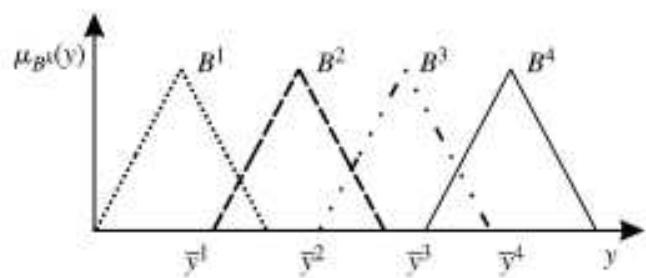
Podstawiając zależność (9.46) do wzoru (9.36), mamy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T \left\{ \begin{array}{l} \max \left\{ \frac{n}{i=1} \{ \mu_{A_i^r}(\bar{x}_i) \}, 1 - \frac{n}{i=1} \{ \mu_{A_i^r}(\bar{x}_i) \} \right\}, \\ \frac{N}{k=1} \left\{ \max \left\{ \min \left[\frac{n}{i=1} \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r) \right], 1 - \frac{n}{i=1} \{ \mu_{A_i^k}(\bar{x}_i) \} \right\} \right\} \end{array} \right\}}{\sum_{r=1}^N T \left\{ \begin{array}{l} \max \left\{ \frac{n}{i=1} \{ \mu_{A_i^r}(\bar{x}_i) \}, 1 - \frac{n}{i=1} \{ \mu_{A_i^r}(\bar{x}_i) \} \right\}, \\ \frac{N}{k=1} \left\{ \max \left\{ \min \left[\frac{n}{i=1} \{ \mu_{A_i^k}(\bar{x}_i) \}, \mu_{B^k}(\bar{y}^r) \right], 1 - \frac{n}{i=1} \{ \mu_{A_i^k}(\bar{x}_i) \} \right\} \right\} \end{array} \right\}}. \quad (9.47)$$

Podstawiając do wzoru (9.47) zależności (9.13) i (9.14), otrzymujemy:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\}, \\ \min \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, \exp \left[- \left(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k} \right)^2 \right] \right\}, \\ \max \left\{ 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \right\} \end{array} \right\}}{\sum_{r=1}^N T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\}, \\ \min \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, \exp \left[- \left(\frac{\bar{y}^r - \bar{y}^k}{\sigma^k} \right)^2 \right] \right\}, \\ \max \left\{ 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \right\} \end{array} \right\}}. \quad (9.48)$$

Przedmiotem badań były również systemy uproszczone, które charakteryzują się niewielkim zachodzeniem na siebie lub całkowitym odseparowaniem jeden od drugiego zbiorów rozmytych wyjściowych B^k . W takiej sytuacji spełniony jest warunek $\mu_{B^k}(\bar{y}^r) \approx 0$, co obrazuje rysunek 9.3.



Rys. 9.3. Przykładowe zbiory rozmyte spełniające założenie $\mu_{B^k}(\bar{y}^r) \approx 0$

Jeśli $\mu_{B^k}(\bar{y}^r) \approx 0$, to z zależności (9.39) otrzymamy uproszczoną strukturę Łukasiewicza postaci

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T \left\{ \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\} \right\}}{\sum_{r=1}^N T \left\{ \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\} \right\}}. \quad (9.49)$$

Podobnie, jeśli $\mu_{B^k}(\bar{y}^r) \approx 0$, z zależności (9.48) otrzymamy uproszczoną strukturę

Zadeha daną wzorem

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\}, \\ \sum_{\substack{k=1 \\ k \neq r}}^N T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \right\} \end{array} \right\} \end{array} \right\}}{\sum_{r=1}^N T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right\}, \\ \sum_{\substack{k=1 \\ k \neq r}}^N T \left\{ \begin{array}{l} \max \left\{ \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, 1 - \frac{n}{T} \left\{ \exp \left[- \left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \right\} \end{array} \right\} \end{array} \right\}}. \quad (9.50)$$

W systemach (9.39), (9.42), (9.45) i (9.48) uczeniu podlegają następujące parametry funkcji przynależności: $\bar{x}_i^k, \sigma_i^k, \bar{y}^k, \sigma^k$. Przedmiotem uczenia w strukturach uproszczonych (9.49) i (9.50) są parametry $\bar{x}_i^k, \sigma_i^k, \bar{y}^k$. Rozwiążemy zagadnienia modelowania za pomocą struktury Łukasiewicza, binarnej, Reichenbacha, Łukasiewicza uproszczonej, Zadeha, Willmotta oraz Zadeha uproszczonej.

9.4.1.1. Polimeryzacja

W tabeli 9.26 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.26. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza	0,0065	5863
Binarna	0,0063	5980
Reichenbacha	0,0040	5494
Łukasiewicza uproszczona	0,0059	3385
Zadeha	0,0038	3648
Willmotta	0,0056	5918
Zadeha uproszczona	0,0049	3432

Jak wynika z tabeli 9.26, najmniejszy błąd wyniósł 0,0038 i został uzyskany dla struktury Zadeha.

W tabeli 9.27 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany. Jak wynika z tej tabeli, nie dla wszystkich struktur udało się osiągnąć zadaną wartość błędu.

Tabela 9.27. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0040
Lukasiewicza	—	—	—
Binarna	—	—	—
Reichenbacha	1627	1903	2783
Łukasiewicza uproszczona	—	—	—
Zadeha	949	1022	1809
Willmotta	—	—	—
Zadeha uproszczona	2844	3432	—

9.4.1.2. HANG

W tabeli 9.28 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.28. Najmniejszy błąd uzyskany w wyniku uczenia

HANG		
Struktura	Najmniejszy błąd	Liczba epok
Lukasiewicza	0,0289	3908
Binarna	0,0177	7773
Reichenbacha	0,0320	7989
Łukasiewicza uproszczona	0,0361	6536
Zadeha	0,0216	5288
Willmotta	0,0366	7327
Zadeha uproszczona	0,0265	2317

Jak wynika z tabeli 9.28, najmniejszy błąd wyniósł 0,0177 i został uzyskany dla struktury binarnej. W tabeli 9.29 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.29. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Lukasiewicza	—	—	—
Binarna	1795	2996	3762
Reichenbacha	—	—	—
Łukasiewicza uproszczona	—	—	—
Zadeha	3382	3875	4218
Willmotta	—	—	—
Zadeha uproszczona	1787	—	—

Jak wynika z tabeli 9.29, nie dla wszystkich struktur udało się otrzymać zadaną wartość błędu.

9.4.1.3. NDP

W tabeli 9.30 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.30. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza	0,0166	457
Binarna	0,0149	437
Reichenbacha	0,0157	454
Łukasiewicza uproszczona	0,0229	497
Zadeha	0,0156	498
Willmotta	0,0180	488
Zadeha uproszczona	0,0156	496

Jak wynika z tabeli 9.30, najmniejszy błąd wyniósł 0,0149 i został uzyskany dla struktury binarnej. W tabeli 9.31 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.31. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Łukasiewicza	255	276	313
Binarna	74	111	166
Reichenbacha	166	173	251
Łukasiewicza uproszczona	190	497	—
Zadeha	38	59	109
Willmotta	121	171	303
Zadeha uproszczona	39	83	147

Jak wynika z tabeli 9.31, dla struktury uproszczonej Łukasiewicza nie udało się uzyskać błędu 0,020.

9.4.1.4. Modelowanie smaku ryżu

W tabeli 9.32 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.32. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza	0,0221	4048
Binarna	0,0230	3201
Reichenbacha	0,0212	4575
Łukasiewicza uproszczona	0,0243	2328
Zadeha	0,0219	4534
Willmotta	0,0211	2605
Zadeha uproszczona	0,0246	4588

Jak wynika z tabeli 9.32, najmniejszy błąd wyniósł 0,0211 i został uzyskany dla struktury Willmotta. W tabeli 9.33 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.33. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,028	0,025	0,022
Łukasiewicza	408	2327	—
Binarna	286	817	—
Reichenbacha	313	938	2354
Łukasiewicza uproszczona	1030	1850	—
Zadeha	344	1484	4534
Willmotta	134	517	2605
Zadeha uproszczona	998	4588	—

Jak wynika z tabeli 9.33, nie dla wszystkich struktur udało się otrzymać zadaną wartość błędu równą 0,022.

9.4.2. Systemy typu M2

Rozważymy systemy typu logicznego, do których konstrukcji zastosowano definicje norm trójkątnych z uwzględnieniem wag w_k , charakteryzujących ważność poszczególnych reguł. Korzystając z definicji ważonej t -normy oraz zależności (9.39), (9.42), (9.45) i (9.48)–(9.50), otrzymujemy następujące systemy neuronowo-rozmyte:

a) System neuronowo-rozmyty z wagami reguł oraz rozmytą implikacją Łukasiewicza

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \sum_{k=1}^N \left\{ \min \left[1, 1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right], w_k \right\}}{\sum_{r=1}^N T^* \sum_{k=1}^N \left\{ \min \left[1, 1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right], w_k \right\}}. \quad (9.51)$$

b) System neuronowo-rozmyty z wagami reguł i rozmytą implikacją binarną

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \sum_{k=1}^N \left\{ \max \left[1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right), \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right], w_k \right\}}{\sum_{r=1}^N T^* \sum_{k=1}^N \left\{ \max \left[1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right), \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right], w_k \right\}}. \quad (9.52)$$

c) System neuronowo-rozmyty z wagami reguł oraz rozmytą implikacją Reichenbacha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \sum_{k=1}^N \left\{ 1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right) \left(1 - \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right), w_k \right\}}{\sum_{r=1}^N T^* \sum_{k=1}^N \left\{ 1 - \frac{n}{i=1} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right) \left(1 - \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right), w_k \right\}}. \quad (9.53)$$

d) System neuronowo-rozmyty z wagami reguł i rozmytą implikacją Zadeha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \left\{ \begin{array}{l} \max \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_r, \\ \frac{N}{T^*} \left\{ \max \left\{ \begin{array}{l} \min \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right\}, \\ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \end{array} \right\}, w_k \end{array} \right\}}{\sum_{r=1}^N T^* \left\{ \begin{array}{l} \max \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_r, \\ \frac{N}{T^*} \left\{ \max \left\{ \begin{array}{l} \min \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right\}, \\ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\} \end{array} \right\}, w_k \end{array} \right\}}. \quad (9.54)$$

e) Uproszczony system neuronowo-rozmyty z wagami reguł i rozmytą implikacją Łukasiewicza

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \sum_{\substack{k=1 \\ k \neq r}}^N \left\{ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_k \right\}}{\sum_{r=1}^N \sum_{\substack{k=1 \\ k \neq r}}^N \left\{ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_k \right\}}. \quad (9.55)$$

f) Uproszczony system neuronowo-rozmyty z wagami reguł i rozmytą implikacją Zadeha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \left\{ \begin{array}{l} \left(\max \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right), w_r, \right. \\ \left. T^* \left\{ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_k \right\} \end{array} \right\}}{\sum_{r=1}^N T^* \left\{ \begin{array}{l} \left(\max \left\{ \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}, 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} \right), w_r, \right. \\ \left. T^* \left\{ 1 - \frac{n}{i=1} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right] \right\}, w_k \right\} \end{array} \right\}}. \quad (9.56)$$

W systemach (9.51)–(9.54) uczeniu podlegają parametry funkcji przynależności $\bar{x}_i^k, \sigma_i^k, \bar{y}^k, \sigma^k$ oraz wagi w_k . Przedmiotem uczenia w systemach (9.55) i (9.56) są parametry $\bar{x}_i^k, \sigma_i^k, \bar{y}^k$ oraz wagi w_k .

9.4.2.1. Polimeryzacja

W tabeli 9.34 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi. Z tabeli tej wynika, że najmniejszy błąd wyniósł 0,0030 i został uzyskany dla struktury Zadeha z wagami reguł.

Tabela 9.34. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Lukasiewicza z wagami reguł	0,0041	4765
Binarna z wagami reguł	0,0054	5980
Reichenbacha z wagami reguł	0,0037	4653
Lukasiewicza uproszczona z wagami reguł	0,0039	4694
Zadeha z wagami reguł	0,0030	5650
Willmotta z wagami reguł	0,0047	5539
Zadeha uproszczona z wagami reguł	0,0041	5151

Tabela 9.35. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,0055	0,0050	0,0040
Lukasiewicza z wagami reguł	1258	1662	3266
Binarna z wagami reguł	5980	—	—
Reichenbacha z wagami reguł	1385	1385	2521
Lukasiewicza uproszczona z wagami reguł	4	4	209
Zadeha z wagami reguł	1497	1497	2726
Willmotta z wagami reguł	1405	3084	—
Zadeha uproszczona z wagami reguł	367	701	3103

W tabeli 9.35 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany. Jak wynika z tej tabeli, nie dla wszystkich struktur udało się otrzymać zadaną wartość błędu.

9.4.2.2. HANG

W tabeli 9.36 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.36. Najmniejszy błąd uzyskany w wyniku uczenia

HANG		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami reguł	0,0247	6500
Binarna z wagami reguł	0,0161	6525
Reichenbacha z wagami reguł	0,0115	7580
Łukasiewicza uproszczona z wagami reguł	0,0350	1840
Zadeha z wagami reguł	0,0202	5290
Willmotta z wagami reguł	0,0335	7977
Zadeha uproszczona z wagami reguł	0,0231	7935

Jak wynika z tabeli 9.36, najmniejszy błąd wyniósł 0,0115 i został uzyskany dla struktury Reichenbacha z wagami reguł. W tabeli 9.37 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.37. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,028	0,026	0,024
Łukasiewicza z wagami reguł	3771	3908	—
Binarna z wagami reguł	1320	1320	1929
Reichenbacha z wagami reguł	506	660	660
Łukasiewicza uproszczona z wagami reguł	—	—	—
Zadeha z wagami reguł	3380	3393	4089
Willmotta z wagami reguł	—	—	—
Zadeha uproszczona z wagami reguł	2483	2483	4139

Jak wynika z tabeli 9.37, nie dla wszystkich struktur udało się otrzymać zadaną wartość błędu.

9.4.2.3. NDP

W tabeli 9.38 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.38. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami reguł	0,0161	492
Binarna z wagami reguł	0,0131	498
Reichenbacha z wagami reguł	0,0140	489
Łukasiewicza uproszczona z wagami reguł	0,0177	459
Zadeha z wagami reguł	0,0148	499
Willmotta z wagami reguł	0,0165	486
Zadeha uproszczona z wagami reguł	0,0142	448

Jak wynika z tabeli 9.38, najmniejszy błąd wyniósł 0,0131 i został uzyskany dla struktury binarnej z wagami reguł. W tabeli 9.39 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.39. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Łukasiewicza z wagami reguł	170	218	274
Binarna z wagami reguł	101	119	151
Reichenbacha z wagami reguł	139	153	186
Łukasiewicza uproszczona z wagami reguł	267	277	364
Zadeha z wagami reguł	222	281	352
Willmotta z wagami reguł	86	121	331
Zadeha uproszczona z wagami reguł	58	78	212

9.4.2.4. Modelowanie smaku ryżu

W tabeli 9.40 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.40. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami reguł	0,0207	3257
Binarna z wagami reguł	0,0219	3897
Reichenbacha z wagami reguł	0,0205	3800
Łukasiewicza uproszczona z wagami reguł	0,0222	2841
Zadeha z wagami reguł	0,0205	3531
Willmotta z wagami reguł	0,0199	3805
Zadeha uproszczona z wagami reguł	0,0227	4432

Jak wynika z tabeli 9.40, najmniejszy błąd wyniósł 0,0199 i został uzyskany dla struktury Willmotta z wagami reguł. W tabeli 9.41 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.41. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Łukasiewicza z wagami reguł	1	16	462
Binarna z wagami reguł	143	1117	3387
Reichenbacha z wagami reguł	38	185	394
Łukasiewicza uproszczona z wagami reguł	397	1152	—
Zadeha z wagami reguł	108	314	879
Willmotta z wagami reguł	22	78	374
Zadeha uproszczona z wagami reguł	461	696	—

Jak wynika z tabeli 9.41, błędu 0,022 nie udało się uzyskać dla uproszczonych struktur Łukasiewicza z wagami reguł i Zadeha z wagami reguł.

9.4.3. Systemy typu M3

Rozważymy systemy typu logicznego, do których konstrukcji zastosowano definicje norm trójkątnych z uwzględnieniem wag w_k , charakteryzujących ważność poszczególnych reguł, oraz wag $w_{i,k}$, charakteryzujących ważność poszczególnych zmiennych lingwistycznych wejściowych. Korzystając z definicji ważonej t -normy oraz zależności (9.37), (9.40), (9.43), (9.46), (9.49) i (9.50), otrzymujemy następujące systemy neuronowo-rozmyte:

a) System neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją Łukasiewicza

$$\bar{y} =$$

$$\frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^{*N} \left\{ \min \left[1, 1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right], w_k \right\} \right\}}{\sum_{r=1}^N T_{k=1}^{*N} \left\{ \min \left[1, 1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right) + \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right], w_k \right\} \right\}}, \quad (9.57)$$

b) System neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją binarną

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^{*N} \left\{ \max \left[1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right), \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right], w_k \right\} \right\}}{\sum_{r=1}^N T_{k=1}^{*N} \left\{ \max \left[1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right), \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right], w_k \right\} \right\}}. \quad (9.58)$$

c) System neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją Reichenbacha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k=1}^{*N} \left\{ 1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right) (1 - \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right]), w_k \right\}}{\sum_{r=1}^N T_{k=1}^{*N} \left\{ 1 - T_{i=1}^{*n} \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right) (1 - \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right]), w_k \right\}}. \quad (9.59)$$

d) System neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją Zadeha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\}, 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\} \right\}, w_r, \\ T_{k=1}^* \left\{ \max \left\{ \begin{array}{l} \min \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right\}, \\ 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\} \end{array} \right\}, w_k \end{array} \right\}}{\sum_{r=1}^N T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\}, 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\} \right\}, w_r, \\ T_{k=1}^* \left\{ \max \left\{ \begin{array}{l} \min \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, \exp \left[-\left(\frac{\bar{y}^r - \bar{y}^k}{\sigma_i^k} \right)^2 \right] \right\}, \\ 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\} \end{array} \right\}, w_k \end{array} \right\}}. \quad (9.60)$$

e) Uproszczony system neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją Łukasiewicza

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T_{k \neq r}^{*N} \left\{ 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, w_k \right\}}{\sum_{r=1}^N T_{k \neq r}^{*N} \left\{ 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, w_k \right\}}. \quad (9.61)$$

f) Uproszczony system neuronowo-rozmyty z wagami wejścia oraz reguł i rozmytą implikacją Zadeha

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\}, 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\} \right\}, w_r, \\ T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, w_k \end{array} \right\} \end{array} \right\}}{\sum_{r=1}^N T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\}, 1 - T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right], w_{i,r} \right\} \right\}, w_r, \\ T^* \left\{ \begin{array}{l} \max \left\{ T_{i=1}^{*n} \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^k}{\sigma_i^k} \right)^2 \right], w_{i,k} \right\}, w_k \end{array} \right\} \end{array} \right\}}. \quad (9.62)$$

W systemach (9.57)–(9.60) przedmiotem uczenia są parametry funkcji przynależności $\bar{x}_i^k, \sigma_i^k, \bar{y}^k, \sigma^k$ oraz wagi $w_{i,k}$ i w_k . Przedmiotem uczenia w systemach (9.61) i (9.62) są parametry $\bar{x}_i^k, \sigma_i^k, \bar{y}^k$ oraz wagi $w_{i,k}$ i w_k . Systemy neuronowo-rozmyte (9.57)–(9.62) zastosowano do rozwiązywania czterech problemów wymienionych w tabeli 9.1.

9.4.3.1. Polimeryzacja

W tabeli 9.42 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.42. Najmniejszy błąd uzyskany w wyniku uczenia

POLIMERYZACJA		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami wejścia i reguł	0,0038	4773
Binarna z wagami wejścia i reguł	0,0036	4896
Reichenbacha z wagami wejścia i reguł	0,0034	4704
Łukasiewicza uproszczona z wagami wejścia i reguł	0,0037	4815
Zadeha z wagami wejścia i reguł	0,0028	5064
Willmotta z wagami wejścia i reguł	0,0039	4810
Zadeha uproszczona z wagami wejścia i reguł	0,0038	5515

Jak wynika z tabeli 9.42, najmniejszy błąd wyniósł 0,0028 i został uzyskany dla struktury Zadeha z wagami wejścia i reguł. W tabeli 9.43 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.43. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	POLIMERYZACJA		
	Wartość błędu		
	0,0055	0,0050	0,0040
Łukasiewicza z wagami wejść i reguł	1	9	867
Binarna z wagami wejść i reguł	2305	2386	2798
Reichenbacha z wagami wejść i reguł	1915	2303	2549
Łukasiewicza uproszczona z wagami wejść i reguł	2502	2821	3225
Zadeha z wagami wejść i reguł	1	1	6
Willmotta z wagami wejść i reguł	11	90	1341
Zadeha uproszczona z wagami wejść i reguł	2	2	206

9.4.3.2. HANG

W tabeli 9.44 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.44. Najmniejszy błąd uzyskany w wyniku uczenia

Struktura	HANG	
	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami wejść i reguł	0,0207	6502
Binarna z wagami wejść i reguł	0,0110	7882
Reichenbacha z wagami wejść i reguł	0,0092	7390
Łukasiewicza uproszczona z wagami wejść i reguł	0,0203	7996
Zadeha z wagami wejść i reguł	0,0105	5533
Willmotta z wagami wejść i reguł	0,0300	6545
Zadeha uproszczona z wagami wejść i reguł	0,0178	8000

Tabela 9.45. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	HANG		
	Wartość błędu		
	0,028	0,026	0,024
Łukasiewicza z wagami wejść i reguł	3724	3771	3771
Binarna z wagami wejść i reguł	556	608	608
Reichenbacha z wagami wejść i reguł	603	678	978
Łukasiewicza uproszczona z wagami wejść i reguł	7992	7992	7992
Zadeha z wagami wejść i reguł	666	666	1115
Willmotta z wagami wejść i reguł	—	—	—
Zadeha uproszczona z wagami wejść i reguł	3943	4408	5407

Jak wynika z tabeli 9.44, najmniejszy błąd wyniósł 0,0092 i został uzyskany dla struktury Reichenbacha z wagami wejść i reguł.

W tabeli 9.45 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany. Jak wynika z tej tabeli, dla struktury Willmotta z wagami wejść i reguł nie udało się uzyskać zadanych poziomów błędów.

9.4.3.3. NDP

W tabeli 9.46 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędowi.

Tabela 9.46. Najmniejszy błąd uzyskany w wyniku uczenia

NDP		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami wejść i reguł	0,0140	498
Binarna z wagami wejść i reguł	0,0121	479
Reichenbacha z wagami wejść i reguł	0,0133	497
Łukasiewicza uproszczona z wagami wejść i reguł	0,0162	457
Zadeha z wagami wejść i reguł	0,0140	4
Willmotta z wagami wejść i reguł	0,0141	496
Zadeha uproszczona z wagami wejść i reguł	0,0135	496

Jak wynika z tabeli 9.46, najmniejszy błąd wyniósł 0,0121 i został uzyskany dla struktury binarnej z wagami wejść i reguł. W tabeli 9.47 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.47. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

Struktura	Wartość błędu		
	0,026	0,023	0,020
Łukasiewicza z wagami wejść i reguł	279	295	368
Binarna z wagami wejść i reguł	61	80	94
Reichenbacha z wagami wejść i reguł	107	176	237
Łukasiewicza uproszczona z wagami wejść i reguł	285	299	315
Zadeha z wagami wejść i reguł	95	109	142
Willmotta z wagami wejść i reguł	80	109	150
Zadeha uproszczona z wagami wejść i reguł	60	82	170

9.4.3.4. Modelowanie smaku ryżu

W tabeli 9.48 przedstawiono najmniejszy błąd dla poszczególnych struktur oraz liczbę epok odpowiadającą temu błędom.

Tabela 9.48. Najmniejszy błąd uzyskany w wyniku uczenia

MODELOWANIE SMAKU RYŻU		
Struktura	Najmniejszy błąd	Liczba epok
Łukasiewicza z wagami wejść i reguł	0,0192	3031
Binarna z wagami wejść i reguł	0,0194	4164
Reichenbacha z wagami wejść i reguł	0,0191	4460
Łukasiewicza uproszczona z wagami wejść i reguł	0,0201	4804
Zadeha z wagami wejść i reguł	0,0164	3994
Willmotta z wagami wejść i reguł	0,0187	3916
Zadeha uproszczona z wagami wejść i reguł	0,0186	3646

Jak wynika z tabeli 9.48, najmniejszy błąd wyniósł 0,0164 i został uzyskany dla struktury Zadeha z wagami wejść i reguł. W tabeli 9.49 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.49. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Łukasiewicza z wagami wejść i reguł	74	331	2045
Binarna z wagami wejść i reguł	143	317	1679
Reichenbacha z wagami wejść i reguł	2	3	8
Łukasiewicza uproszczona z wagami wejść i reguł	165	450	1702
Zadeha z wagami wejść i reguł	40	143	197
Willmotta z wagami wejść i reguł	1	1	37
Zadeha uproszczona z wagami wejść i reguł	76	202	404

9.5. Systemy neuronowo-rozmyte typu Takagi–Sugeno

W modelu rozmytym typu Takagi–Sugeno [246] baza reguł ma charakter rozmyty tylko w części **JEŻELI**, natomiast w części **TO** występują zależności funkcyjne

$$R^{(r)} : \text{JEŻELI } (x_1 \text{ jest } A'_1 \text{ I } x_2 \text{ jest } A'_2 \dots \text{ I } x_n \text{ jest } A'_n) \text{ TO } y_1 = f^{(r)}(x_1, x_2, \dots, x_n) \quad (9.63)$$

Jeżeli założymy, że wejściem systemu rozmytego jest sygnał $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$, to aby otrzymać sygnał wyjściowy \bar{y} systemu, najpierw wyznaczamy

$$T(\mu_{A'_1}(\bar{x}_1), \mu_{A'_2}(\bar{x}_2), \dots, \mu_{A'_n}(\bar{x}_n)), \quad r = 1, \dots, N. \quad (9.64)$$

Następnym krokiem jest obliczenie

$$\bar{y}_r = f^{(r)}(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n), \quad r = 1, \dots, N. \quad (9.65)$$

Sygnał wyjściowy systemu rozmytego Takagi–Sugeno jest znormalizowaną sumą ważoną poszczególnych wyjść $\bar{y}_1, \dots, \bar{y}_N$, tzn.

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}_r T_{i=1}^n \{\mu_{A'_i}(\bar{x}_i)\}}{\sum_{r=1}^N T_{i=1}^n \{\mu_{A'_i}(\bar{x}_i)\}}. \quad (9.66)$$

W dalszej części tego podrozdziału rozważymy systemy Takagi–Sugeno z liniowymi zależnościami w następnikach bazy reguł, tzn.

$$\begin{aligned} R^{(r)} : & \text{JEŻELI } (x_1 \text{ jest } A'_1 \text{ I } x_2 \text{ jest } A'_2 \dots \text{ I } x_n \text{ jest } A'_n) \\ & \text{TO } y_r = c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n \end{aligned} \quad (9.67)$$

dla $r = 1, \dots, N$. Warto zauważyć, że jeżeli $c_i^{(r)} = 0$, $i = 1, \dots, n$, to system (9.66) redukuje się do uproszczonego systemu Mamdaniego danego wzorem (9.12) i wówczas $c_0^{(r)} = \bar{y}^r$, $r = 1, \dots, N$.

Systemy typu Takagi–Sugeno zostały zastosowane do rozwiązania zadań aproksymacji i identyfikacji (polimeryzacja, HANG, NDP, modelowanie smaku ryżu). Podobnie jak w przypadku struktur typu Mamdaniego i logicznego rozważymy trzy rodzaje systemów, tzn. bez wag, z wagami reguł oraz z wagami reguł i wagami wejść odzwierciedlających ważność poszczególnych zmiennych lingwistycznych.

9.5.1. Systemy typu M1

Do konstrukcji systemu neuronowo-rozmytego wykorzystano gaussowskie funkcje przynależności oraz założenie, że poprzedniki w każdej regule są połączone poprzez t -normę typu iloczyn. W takiej sytuacji zależność (9.66) przybiera postać

$$\begin{aligned} \bar{y} &= \frac{\sum_{r=1}^N T_{i=1}^n \{\mu_{A'_i}(\bar{x}_i)\} (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N T_{i=1}^n \{\mu_{A'_i}(\bar{x}_i)\}} \\ &= \frac{\sum_{r=1}^N T_{i=1}^n \{\exp\left[-\left(\frac{\bar{x}_i - \bar{x}'_i}{\sigma'_i}\right)^2\right]\} (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N T_{i=1}^n \{\exp\left[-\left(\frac{\bar{x}_i - \bar{x}'_i}{\sigma'_i}\right)^2\right]\}} \\ &= \frac{\sum_{r=1}^N \left[\prod_{i=1}^n \left(\exp\left[-\left(\frac{\bar{x}_i - \bar{x}'_i}{\sigma'_i}\right)^2\right] \right) \right] (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N \left[\prod_{i=1}^n \left(\exp\left[-\left(\frac{\bar{x}_i - \bar{x}'_i}{\sigma'_i}\right)^2\right] \right) \right]} \end{aligned} \quad (9.68)$$

Metodą wstecznej propagacji błędów uczone wszystkie parametry systemu neuronowo-rozmytego: uczone środki i szerokości funkcji gaussowskich oraz parametry funkcji.

9.5.1.1. Polimeryzacja

Najmniejszy błąd dla struktury typu Takagi–Sugeno wyniósł 0,0034 i został uzyskany w 3430. epoce. W tabeli 9.50 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.50. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Takagi–Sugeno	72	83	83

9.5.1.2. HANG

Najmniejszy błąd dla struktury typu Takagi–Sugeno wyniósł 0,0197 i został uzyskany w 7551. epoce. W tabeli 9.51 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.51. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Takagi–Sugeno	4280	5159	5593

9.5.1.3. NDP

Najmniejszy błąd dla struktury typu Takagi–Sugeno wyniósł 0,0156 i został uzyskany w 481. epoce. W tabeli 9.52 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.52. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Takagi–Sugeno	36	66	122

9.5.1.4. Modelowanie smaku ryżu

Najmniejszy błąd dla struktury typu Takagi–Sugeno wyniósł 0,0176 i został uzyskany w 1264. epoce. W tabeli 9.53 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.53. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Takagi–Sugeno	546	1060	1951

9.5.2. Systemy typu M2

Wprowadzając do systemu (9.68) wagę określającą ważność poszczególnych reguł otrzymujemy następującą zależność:

$$\begin{aligned}\bar{y} &= \frac{\sum_{r=1}^N w_r T_{i=1}^n \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\} (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N w_r T_{i=1}^n \left\{ \exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right\}} \\ &= \frac{\sum_{r=1}^N w_r \left[\prod_{i=1}^n \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right) \right] (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N w_r \left[\prod_{i=1}^n \left(\exp \left[-\left(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r} \right)^2 \right] \right) \right]}\end{aligned}\quad (9.69)$$

Uczemu za pomocą metody wstecznej propagacji błędów poddano wszystkie parametry systemu neuronowo-rozmytego: uczono środki i szerokości funkcji gaussowskich, wagi reguł oraz parametry funkcji.

9.5.2.1. Polimeryzacja

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami reguł wyniósł 0,0031 i został uzyskany w 3098. epoce. W tabeli 9.54 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.54. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Takagi–Sugeno z wagami reguł	56	57	95

9.5.2.2. HANG

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami reguł wyniósł 0,0145 i został uzyskany w 3008. epoce. W tabeli 9.55 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.55. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Takagi–Sugeno z wagami reguł	478	779	1132

9.5.2.3. NDP

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami reguł wyniósł 0,0140 i został uzyskany w 497. epoce. W tabeli 9.56 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.56. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Takagi–Sugeno z wagami reguł	20	40	79

9.5.2.4. Modelowanie smaku ryżu

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami reguł wyniósł 0,0149 i został uzyskany w 1620. epoce. W tabeli 9.57 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po których błąd ten został uzyskany.

Tabela 9.57. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Takagi–Sugeno z wagami reguł	8	35	67

9.5.3. Systemy typu M3

Wprowadzając do systemu (9.69) wagi określające ważność poszczególnych zmiennych lingwistycznych w każdej regule, otrzymujemy następującą zależność:

$$\begin{aligned}
 \bar{y} &= \frac{\sum_{r=1}^N w_r [T_{i=1}^n \{1 - w_{i,r}(1 - \mu_{A_i^r}(\bar{x}_i))\}] (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N w_r [T_{i=1}^n \{1 - w_{i,r}(1 - \mu_{A_i^r}(\bar{x}_i))\}]} \\
 &= \frac{\sum_{r=1}^N w_r [T_{i=1}^n \{1 - w_{i,r}(1 - \exp[-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2]\})] (c_0^{(r)} + c_1^{(r)} x_1 + \dots + c_n^{(r)} x_n)}{\sum_{r=1}^N w_r [T_{i=1}^n \{1 - w_{i,r}(1 - \exp[-(\frac{\bar{x}_i - \bar{x}_i^r}{\sigma_i^r})^2]\})]}.
 \end{aligned} \tag{9.70}$$

Uczniu za pomocą metody wstecznej propagacji błędów poddano wszystkie parametry systemu neuronowo-rozmytego: uczono środki i szerokości funkcji gaussowskich, wagi wejść i reguł oraz parametry funkcji.

9.5.3.1. Polimeryzacja

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami wejść i reguł wyniósł 0,0030 i został uzyskany w 4859. epoce. W tabeli 9.58 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.58. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

POLIMERYZACJA			
Struktura	Wartość błędu		
	0,0055	0,0050	0,0045
Takagi–Sugeno z wagami wejść i reguł	36	110	324

9.5.3.2. HANG

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami wejść i reguł wyniósł 0,0116 i został uzyskany w 2381. epoce. W tabeli 9.59 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.59. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

HANG			
Struktura	Wartość błędu		
	0,028	0,026	0,024
Takagi–Sugeno z wagami wejść i reguł	265	465	478

9.5.3.3. NDP

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami wejść i reguł wyniósł 0,0085 i został uzyskany w 495. epoce. W tabeli 9.60 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.60. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

NDP			
Struktura	Wartość błędu		
	0,026	0,023	0,020
Takagi–Sugeno z wagami wejść i reguł	60	77	111

9.5.3.4. Modelowanie smaku ryżu

Najmniejszy błąd dla struktury typu Takagi–Sugeno z wagami wejść i reguł wyniósł 0,0129 i został uzyskany w 4008. epoce. W tabeli 9.61 przedstawiono trzy zadane wartości błędu oraz liczbę epok, po której błąd ten został uzyskany.

Tabela 9.61. Liczba epok wymagana do nauczenia systemu charakteryzującego się określonym błędem

MODELOWANIE SMAKU RYŻU			
Struktura	Wartość błędu		
	0,028	0,025	0,022
Takagi–Sugeno z wagami wejść i reguł	1	1	1

9.6. Algorytmy uczenia systemów neuronowo-rozmytych

W podrozdziałach 9.3, 9.4 i 9.5 kolejno omówiliśmy systemy neuronowo-rozmyte typu Takagi–Sugeno, Mamdaniego i logicznego. W tym podrozdziale wyprowadzimy algorytmy uczenia wymienionych systemów. Były one stosowane w zamieszczonych wcześniej przykładach symulacyjnych (podrozdz. 9.3–9.5).

Wykorzystamy ideę metod wstecznej propagacji błędów, będącej podstawową metodą uczenia sieci neuronowych. Uczenie systemów neuronowo-rozmytych sprowadzimy do zastosowania algorytmów gradientowych, minimalizujących odpowiednio sformułowane kryterium jakości. Przez $\bar{x}(t) \in \mathbf{R}^n$ i $d(t) \in \mathbf{R}$ oznaczmy, odpowiednio, ciąg sygnałów wejściowych i wzorcowych, tzn. zadanych na wyjściu systemu neuronowo-rozmytego. Zagadnienie uczenia tych systemów sprowadza się do wyznaczenia na podstawie ciągu uczącego

$$(\bar{x}(1), d(1)), (\bar{x}(2), d(2)), \dots \quad (9.71)$$

wszystkich parametrów funkcji przynależności oraz wag (wag opisujących ważność reguł oraz ważność poszczególnych zmiennych lingwistycznych w każdej regule), tak aby minimalizować kryterium

$$Q(t) = \frac{1}{2} [f(\bar{x}(t)) - d(t)]^2, \quad (9.72)$$

gdzie

$$\bar{y} = f(\bar{x}(t)) \quad (9.73)$$

jest wyjściem systemu neuronowo-rozmytego typu Mamdaniego, logicznego lub Takagi–Sugeno przedstawionego w poprzednich podrozdziałach. Na przykład w systemach typu Mamdaniego i logicznego parametr \bar{y}^r , $r = 1, \dots, N$, można wyznaczyć za pomocą algorytmu gradientowego

$$\bar{y}^r(t+1) = \bar{y}^r(t) - \eta \frac{\partial Q(t)}{\partial \bar{y}^r(t)}. \quad (9.74)$$

Bezpośrednie wyznaczenie gradientu $\frac{\partial Q(t)}{\partial \bar{y}^r(t)}$ w powyższej procedurze jest skomplikowane z obliczeniowego punktu widzenia. Dlatego wykorzystano analogię między sieciami neuronowymi i neuronowo-rozmytymi, zauważając, że te ostatnie również mają strukturę wielowarstwową. Zatem można zastosować ideę metody wstecznej propagacji błędów do uczenia sieci neuronowo-rozmytych. Przyjęta w tym podrozdziale notacja wyjaśniona

zostanie na przykładzie pojedynczego neuronu opisanego wzorem

$$y = f(s), \quad s = \sum_{i=0}^n x_i w_i, \quad (9.75)$$

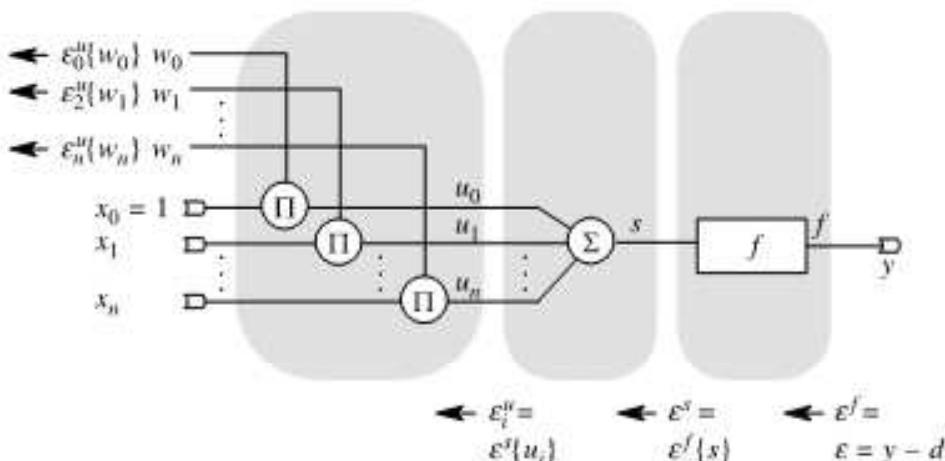
w którym f jest funkcją sigmoidalną, x_i oraz w_i , $i = 0, \dots, n$ są wejściami i wagami neuronu. Niech d będzie sygnałem zadanym na wyjściu neuronu. Wówczas

$$\varepsilon^f = \varepsilon = y - d \quad (9.76)$$

jest błędem na wyjściu neuronu, natomiast wyrażenie

$$\varepsilon^s = \varepsilon^f \{s\} = \varepsilon^f \frac{\partial f(s)}{\partial s} = (y - d) f'(s) \quad (9.77)$$

opisuje błąd propagowany z bloku funkcyjnego f do bloku sumacyjnego s . Na rysunku 9.4 przedstawiono przepływ sygnałów i błędów w pojedynczym neuronie.



Rys. 9.4. Przepływ sygnałów i błędów w pojedynczym neuronie

Najpierw wyprowadzony zostanie algorytm uczenia dla systemu Takagi–Sugeno, a następnie dla systemu typu Mamdaniego i logicznego. Zmodyfikujemy dotychczas stosowane oznaczenia w opisach tych systemów, aby klarownie przedstawić przepływ błędów przez poszczególne bloki wymienionych systemów. Sygnał wyjściowy systemu Takagi–Sugeno można opisać następująco:

$$\bar{y} = \frac{\sum_{r=1}^N w_r^{\text{def}} \cdot T^* \left\{ \frac{\mu_{A'_1}(\bar{x}_1), \mu_{A'_2}(\bar{x}_2), \dots, \mu_{A'_n}(\bar{x}_n);}{w_{1,r}^{\tau}, w_{2,r}^{\tau}, \dots, w_{n,r}^{\tau}} \right\} \cdot (c_{0,r}^f + \sum_{i=1}^n c_{i,r}^f \cdot \bar{x}_i)}{\sum_{r=1}^N w_r^{\text{def}} \cdot T^* \left\{ \frac{\mu_{A'_1}(\bar{x}_1), \mu_{A'_2}(\bar{x}_2), \dots, \mu_{A'_n}(\bar{x}_n);}{w_{1,r}^{\tau}, w_{2,r}^{\tau}, \dots, w_{n,r}^{\tau}} \right\}}, \quad (9.78)$$

przy czym $w_{i,r}^{\tau} \in [0, 1]$, $i = 1, 2, \dots, n$, $r = 1, 2, \dots, N$, oznaczają wagę przesłanek reguł oraz $w_r^{\text{def}} \in [0, 1]$, $r = 1, 2, \dots, N$, oznaczają wagę reguł. Podstawiając

$$T^* \left\{ \frac{\mu_{A'_1}(\bar{x}_1), \mu_{A'_2}(\bar{x}_2), \dots, \mu_{A'_n}(\bar{x}_n);}{w_{1,r}^{\tau}, w_{2,r}^{\tau}, \dots, w_{n,r}^{\tau}} \right\} = \tau_r(\bar{x}) \quad (9.79)$$

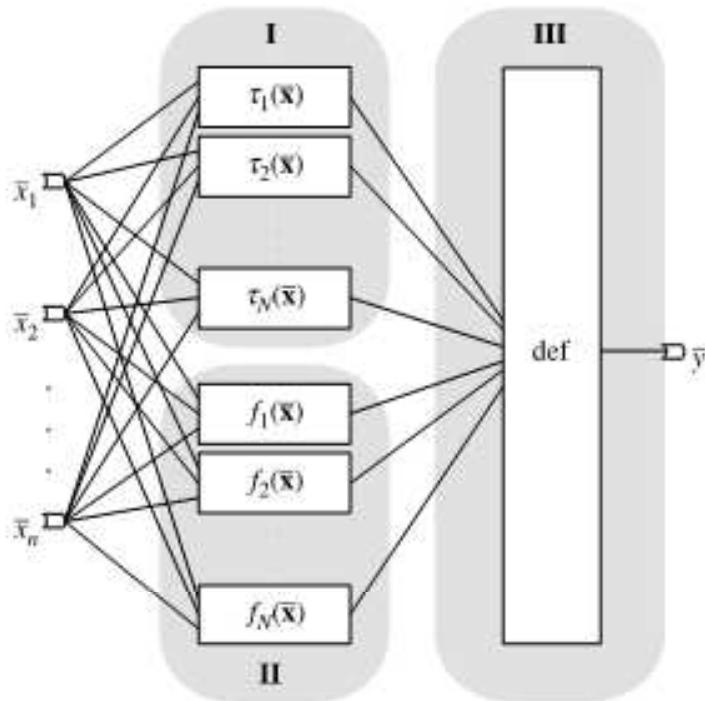
oraz

$$c_{0,r}^f + \sum_{i=1}^n c_{i,r}^f \cdot \bar{x}_i = f_r(\bar{\mathbf{x}}), \quad (9.80)$$

otrzymujemy

$$\bar{y} = \frac{\sum_{r=1}^N w_r^{\text{def}} \cdot \tau_r(\bar{\mathbf{x}}) \cdot f_r(\bar{\mathbf{x}})}{\sum_{r=1}^N w_r^{\text{def}} \cdot \tau_r(\bar{\mathbf{x}})} = \text{def} \left(\frac{\tau_1(\bar{\mathbf{x}}), \dots, \tau_N(\bar{\mathbf{x}}),}{f_1(\bar{\mathbf{x}}), \dots, f_N(\bar{\mathbf{x}});} \frac{w_1^{\text{def}}, \dots, w_N^{\text{def}}}{\text{def}} \right). \quad (9.81)$$

Strukturę sieciową systemu Takagi–Sugeno przedstawia rysunek 9.5.



Rys. 9.5. Struktura sieciowa systemu Takagi–Sugeno

W systemie Takagi–Sugeno uczeniu podlegają następujące parametry:

- $p_{u,i,r}^A$, $u = 1, 2, \dots, P^A$, parametry funkcji przynależności wejściowych zbiorów rozmytych,
- $c_{i,r}^f$, $i = 0, 1, \dots, n$, $r = 1, 2, \dots, N$, parametry bloków funkcyjnych,
- $w_{i,r}^{\tau}$, $i = 1, 2, \dots, n$, $r = 1, 2, \dots, N$, wagi przesłanek,
- w_r^{def} , $r = 1, 2, \dots, N$, wagi reguł.

Parametry systemu Takagi–Sugeno modyfikuje się w sposób iteracyjny zgodnie z poniższymi zależnościami:

$$p_{u,i,r}^A(t+1) = p_{u,i,r}^A(t) - \eta \Delta p_{u,i,r}^A(t), \quad (9.82)$$

$$w_{i,r}^{\tau}(t+1) = w_{i,r}^{\tau}(t) - \eta \Delta w_{i,r}^{\tau}(t), \quad (9.83)$$

$$c_{0,r}^f(t+1) = c_{0,r}^f(t) - \eta \Delta c_{0,r}^f(t), \quad (9.84)$$

$$c_{i,r}^f(t+1) = c_{i,r}^f(t) - \eta \Delta c_{i,r}^f(t), \quad i = 1, \dots, n, \quad (9.85)$$

$$w_r^{\text{def}}(t+1) = w_r^{\text{def}}(t) - \eta \Delta w_r^{\text{def}}(t). \quad (9.86)$$

Człony Δ w powyższych zależnościach określa się następująco:

$$\Delta p_{u,i,r}^A(t) = \varepsilon_r^\tau \{ p_{u,i,r}^A \}, \quad (9.87)$$

$$\Delta w_{i,r}^\tau(t) = \varepsilon_r^\tau \{ w_{i,r}^\tau \}, \quad (9.88)$$

$$\Delta c_{0,r}^f(t) = \varepsilon_r^f \{ c_{0,r}^f \}, \quad (9.89)$$

$$\Delta c_{i,r}^f(t) = \varepsilon_r^f \{ c_{i,r}^f \}, \quad (9.90)$$

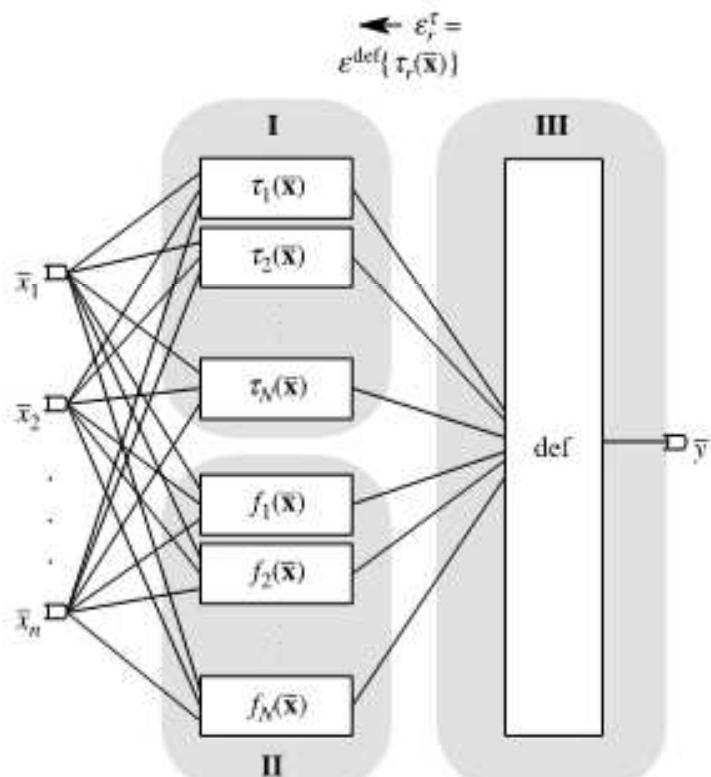
$$\Delta w_r^{\text{def}}(t) = \varepsilon^{\text{def}} \{ w_r^{\text{def}} \}. \quad (9.91)$$

Błędy propagowane przez poszczególne warstwy systemu Takagi–Sugeno wyznacza się następująco (rys. 9.6):

$$\varepsilon_r^\tau = \varepsilon^{\text{def}} \{ \tau_r(\bar{x}) \}, \quad (9.92)$$

$$\varepsilon_r^f = \varepsilon^{\text{def}} \{ f_r(\bar{x}) \}, \quad (9.93)$$

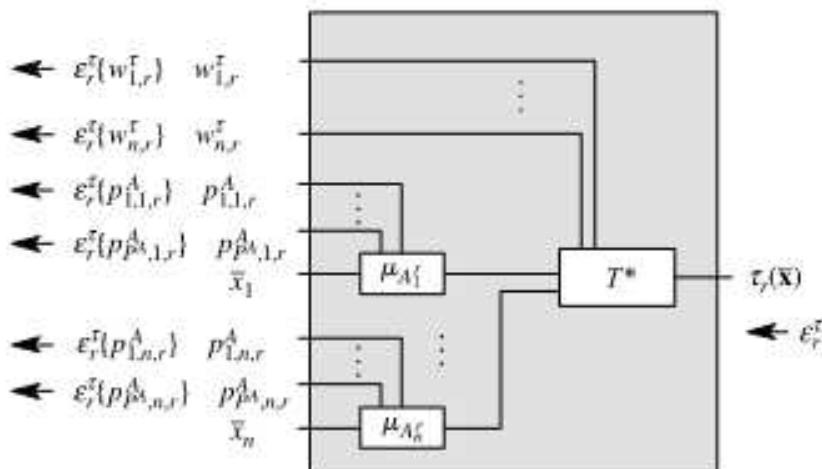
$$\varepsilon^{\text{def}} = \varepsilon = \bar{y} - d. \quad (9.94)$$



Rys. 9.6. Przepływ błędów w systemie Takagi–Sugeno

Błędy propagowane przez bloki poziomów aktywności reguł systemu Takagi–Sugeno określa się następująco (rys. 9.7):

$$\varepsilon_r^\tau \{ p_{u,i,r}^A \} = \varepsilon_r^\tau \frac{\partial T^* \left\{ \frac{\mu_{A'_i}(\bar{x}_1), \mu_{A'_i}(\bar{x}_2), \dots, \mu_{A'_i}(\bar{x}_n);}{w_{1,r}^\tau, w_{2,r}^\tau, \dots, w_{n,r}^\tau} \right\}}{\partial \mu_{A'_i}(\bar{x}_i)} \frac{\partial \mu_{A'_i}(\bar{x}_i)}{\partial p_{u,i,r}^A}, \quad (9.95)$$



Rys. 9.7. Blok poziomu aktywności reguły systemu Takagi–Sugeno

$$\varepsilon_r^T\{w_{i,r}^T\} = \varepsilon_r^T \frac{\partial T^* \left\{ \begin{array}{c} \mu_{A'_1}(\bar{x}_1), \mu_{A'_2}(\bar{x}_2), \dots, \mu_{A'_n}(\bar{x}_n); \\ w_{1,r}^T, w_{2,r}^T, \dots, w_{n,r}^T \end{array} \right\}}{\partial w_{i,r}^T}. \quad (9.96)$$

Należy zwrócić uwagę na fakt, że rozwiązuje się zadanie optymalizacji z ograniczeniami. Dlatego w dalszych rozważaniach wykorzystamy tzw. *funkcję zakresową* $f_z(\cdot)$ daną zależnością

$$f_z(x) = \frac{1}{1 + \exp(-(p_1 x - p_2))}, \quad (9.97)$$

przy czym

$$\frac{\partial f_z(x)}{\partial x} = p_1(1 - f_z(x))f_z(x). \quad (9.98)$$

W symulacjach przyjęto $p_1 = 10$ oraz $p_2 = 5$.

Przykład 9.1

Pokażemy teraz sposób wyznaczenia pochodnych cząstkowych we wzorach (9.95) i (9.96). Wprowadzając zapis funkcji zakresowej do definicji ważonej t -normy, otrzymujemy

$$T^* \left\{ \begin{array}{c} a_1, a_2, \dots, a_n; \\ w_1, w_2, \dots, w_n \end{array} \right\} = T^* \{ \mathbf{a}; \mathbf{w} \} = T_{i=1}^n \{ 1 - f_z(w_i)(1 - a_i) \}. \quad (9.99)$$

W przypadku t -normy algebraicznej mamy

$$T^* \{ \mathbf{a}; \mathbf{w} \} = \prod_{i=1}^n (1 - f_z(w_i)(1 - a_i)). \quad (9.100)$$

Wówczas

$$\frac{\partial T^* \{ \mathbf{a}; \mathbf{w} \}}{\partial a_i} = f_z(w_i) \prod_{\substack{u=1 \\ u \neq i}}^n (1 - f_z(w_u)(1 - a_u)) \quad (9.101)$$

oraz

$$\frac{\partial T^* \{ \mathbf{a}; \mathbf{w} \}}{\partial w_i} = -(1 - a_i) \frac{\partial f_z(w_i)}{\partial w_i} \prod_{\substack{u=1 \\ u \neq i}}^n (1 - f_z(w_u)(1 - a_u)). \quad (9.102)$$

Przykład 9.2

Wyznaczmy pochodne cząstkowe gaussowskiej funkcji przynależności wejściowego zbioru rozmytego A (dla przejrzystości zapisu pomijamy odpowiednie indeksy)

$$\mu_A(x) = \exp\left(-\left(\frac{x - \bar{x}}{\sigma}\right)^2\right). \quad (9.103)$$

Zauważmy, że

$$P^A = 2, \quad p_{1,i,r}^A = x, \quad p_{2,i,r}^A = \sigma. \quad (9.104)$$

Odpowiednie pochodne przybierają postać

$$\frac{\partial \mu_A(x)}{\partial x} = -\mu_A(x) \frac{2(x - \bar{x})}{\sigma^2}, \quad (9.105)$$

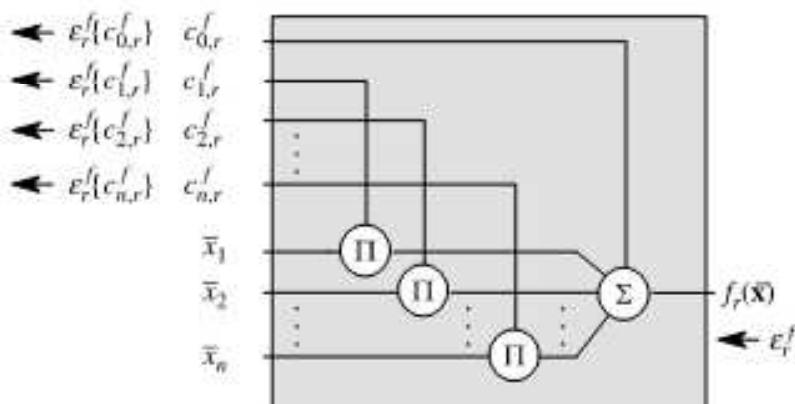
$$\frac{\partial \mu_A(x)}{\partial \bar{x}} = \mu_A(x) \frac{2(x - \bar{x})}{\sigma^2}, \quad (9.106)$$

$$\frac{\partial \mu_A(x)}{\partial \sigma} = \mu_A(x) \frac{2(x - \bar{x})^2}{\sigma^3}. \quad (9.107)$$

Błędy propagowane przez bloki funkcyjne systemu Takagi–Sugeno określają się następująco (rys. 9.8):

$$\varepsilon_r^f \{c_{0,r}^f\} = \varepsilon_r^f, \quad (9.108)$$

$$\varepsilon_r^f \{c_{i,r}^f\} = \varepsilon_r^f \bar{x}_i. \quad (9.109)$$



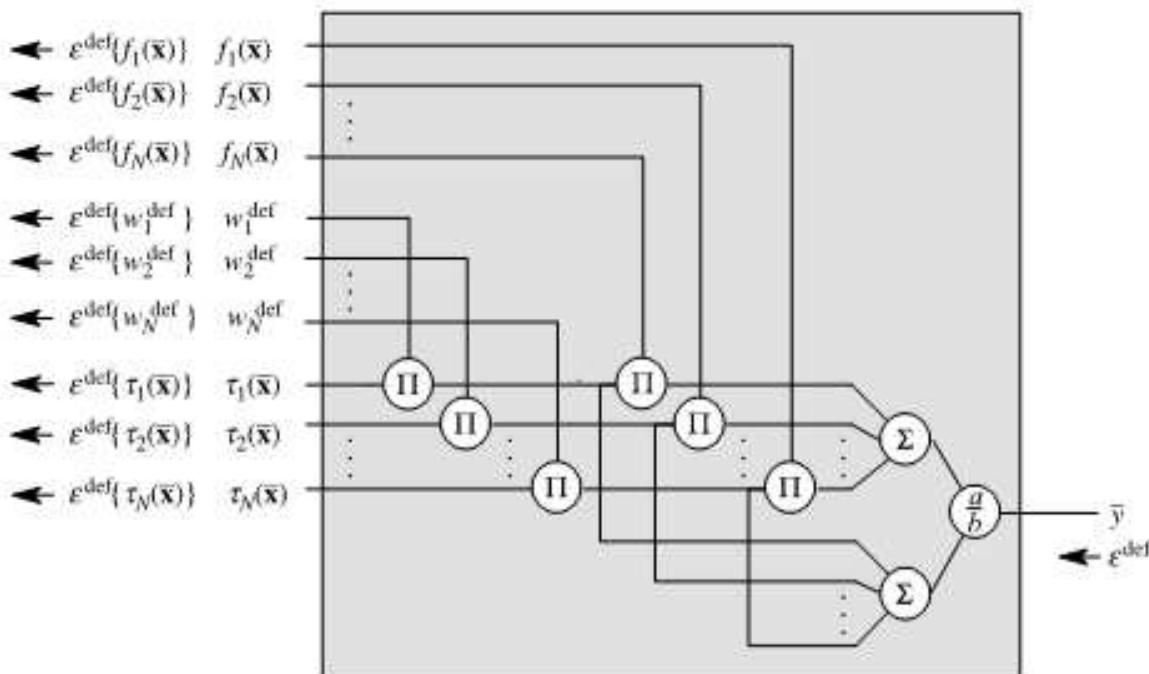
Rys. 9.8. Blok funkcyjny systemu Takagi–Sugeno

Błędy propagowane przez blok wyostrzania systemu Takagi–Sugeno wyznacza się następująco (rys. 9.9):

$$\varepsilon^{\text{def}} \{\tau_r(\bar{x})\} = \varepsilon^{\text{def}} \frac{\partial}{\partial \tau_r(\bar{x})} \text{def} \left(\begin{array}{c} \tau_1(\bar{x}), \dots, \tau_N(\bar{x}), \\ f_1(\bar{x}), \dots, f_N(\bar{x}); \\ w_1^{\text{def}}, \dots, w_N^{\text{def}} \end{array} \right), \quad (9.110)$$

$$\varepsilon^{\text{def}} \{f_r(\bar{x})\} = \varepsilon^{\text{def}} \frac{\partial}{\partial f_r(\bar{x})} \text{def} \left(\begin{array}{c} \tau_1(\bar{x}), \dots, \tau_N(\bar{x}), \\ f_1(\bar{x}), \dots, f_N(\bar{x}); \\ w_1^{\text{def}}, \dots, w_N^{\text{def}} \end{array} \right), \quad (9.111)$$

$$\varepsilon^{\text{def}} \{w_r^{\text{def}}\} = \varepsilon^{\text{def}} \frac{\partial}{\partial w_r^{\text{def}}} \text{def} \left(\begin{array}{c} \tau_1(\bar{x}), \dots, \tau_N(\bar{x}), \\ f_1(\bar{x}), \dots, f_N(\bar{x}); \\ w_1^{\text{def}}, \dots, w_N^{\text{def}} \end{array} \right), \quad (9.112)$$



Rys. 9.9. Blok wyostrzania systemu Takagi-Sugeno

przy czym

$$\text{def} \begin{pmatrix} a_1, a_2, \dots, a_n, \\ b_1, b_2, \dots, b_n; \\ w_1, w_2, \dots, w_n \end{pmatrix} = \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w}) = \frac{\sum_{i=1}^n w_i a_i b_i}{\sum_{i=1}^n w_i a_i}, \quad (9.113)$$

$$\frac{\partial \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w})}{\partial a_j} = (b_j - \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w})) \frac{w_j}{\sum_{i=1}^n w_i a_i}, \quad (9.114)$$

$$\frac{\partial \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w})}{\partial w_j} = (b_j - \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w})) \frac{a_j}{\sum_{i=1}^n w_i a_i}, \quad (9.115)$$

$$\frac{\partial \text{def}(\mathbf{a}, \mathbf{b}; \mathbf{w})}{\partial b_j} = \frac{w_j a_j}{\sum_{i=1}^n w_i a_i}. \quad (9.116)$$

Wyprowadzimy teraz algorytmy uczenia systemów neuronowo-rozmytych typu Mamdaniego i logicznego. Rozważania rozpocznijmy od uogólnionego modelu, który opisuje oba typy systemów. Sygnał wyjściowy takiego systemu można przedstawić następująco:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot \text{agr}_r(\bar{x}, \bar{y}^r)}{\sum_{r=1}^N \text{agr}_r(\bar{x}, \bar{y}^r)}. \quad (9.117)$$

Działanie operatorów $\text{agr}_r(\bar{x}, \bar{y}^r)$, $r = 1, 2, \dots, N$, uzależnione jest od rodzaju wnioskowania stosowanego w danym typie systemu, tzn.

$$\text{agr}_r(\bar{x}, \bar{y}^r) = \begin{cases} S^* \left\{ \frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{\text{agr}}, \dots, w_N^{\text{agr}}}; \right\} & \text{dla wnioskowania Mamdaniego,} \\ T^* \left\{ \frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{\text{agr}}, \dots, w_N^{\text{agr}}}; \right\} & \text{dla wnioskowania logicznego,} \end{cases} \quad (9.118)$$

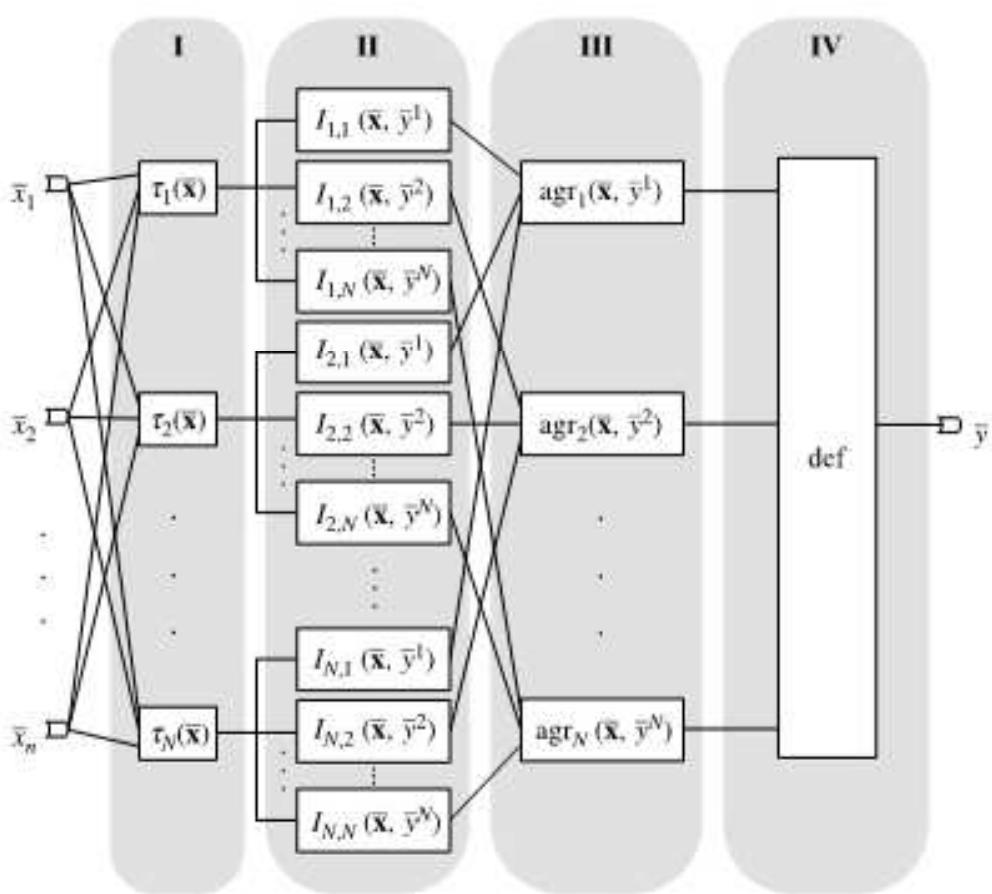
przy czym

$$I_{k,r}(\bar{x}, \bar{y}^r) = \begin{cases} T\{\tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r)\} & \text{dla wnioskowania Mamdaniego,} \\ I_{\text{fuzzy}}(\tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r)) & \text{dla wnioskowania logicznego} \end{cases} \quad (9.119)$$

oraz

$$I_{\text{fuzzy}}(\tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r)) = \begin{cases} S\{N(\tau_k(\bar{x})), \mu_{B^k}(\bar{y}^r)\} & \text{dla } S\text{-implikacji,} \\ t_{\text{mul}}^{-1}\left(\min\left\{1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}\right\}\right) & \text{dla } R\text{-implikacji,} \\ S\{N(\tau_k(\bar{x})), T\{\tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r)\}\} & \text{dla } Q\text{-implikacji.} \end{cases} \quad (9.120)$$

We wzorze (9.120) wykorzystano definicję R -implikacji z uwzględnieniem multiplikatywnych generatorów $t_{\text{mul}}(\cdot)$ archimedesowej t -normy. Operator poziomu aktywności reguły $\tau_k(\bar{x})$, $k = 1, 2, \dots, N$, określony został analogicznie jak dla opisanego wcześniej systemu Takagi–Sugeno. Na rysunku 9.10 przedstawiono strukturę sieciową uogólnionego systemu neuronowo-rozmytego.



Rys. 9.10. Struktura sieciowa systemu neuronowo-rozmytego

W rozważanym przez nas systemie neuronowo-rozmytym uczeniu podlegają następujące parametry:

- $p_{u,i,k}^A$, $u = 1, 2, \dots, P^A$, $i = 1, 2, \dots, n$, $k = 1, 2, \dots, N$, parametry funkcji przynależności wejściowych zbiorów rozmytych,
- $p_{1,k}^B = \bar{y}^k$, $k = 1, 2, \dots, N$, centra funkcji przynależności wyjściowych zbiorów rozmytych,

- $p_{u,k}^B$, $u = 2, 3, \dots, P^B$, $k = 1, 2, \dots, N$, pozostałe parametry funkcji przynależności wyjściowych zbiorów rozmytych,
- $w_{i,k}^\tau$, $i = 1, 2, \dots, n$, $k = 1, 2, \dots, N$, wagi przesłanek,
- w_k^{agr} , $k = 1, 2, \dots, N$, wagi reguł.

Parametry systemu modyfikuje się w sposób iteracyjny zgodnie z poniższymi zależnościami:

$$p_{u,i,k}^A(t+1) = p_{u,i,k}^A(t) - \eta \Delta p_{u,i,k}^A(t), \quad (9.121)$$

$$w_{i,k}^\tau(t+1) = w_{i,k}^\tau(t) - \eta \Delta w_{i,k}^\tau(t), \quad (9.122)$$

$$p_{u,k}^B(t+1) = p_{u,k}^B(t) - \eta \Delta p_{u,k}^B(t), \quad u = 2, \dots, P^B, \quad (9.123)$$

$$\bar{y}^r(t+1) = p_{1,r}^B(t+1) = \bar{y}^r(t) - \eta \Delta \bar{y}^r(t), \quad (9.124)$$

$$w_k^{\text{agr}}(t+1) = w_k^{\text{agr}}(t) - \eta \Delta w_k^{\text{agr}}(t). \quad (9.125)$$

Człony Δ w tych zależnościach określa się następująco:

$$\Delta p_{u,i,k}^A = \varepsilon_k^\tau \{ p_{u,i,k}^A \}, \quad (9.126)$$

$$\Delta w_{i,k}^\tau = \varepsilon_k^\tau \{ w_{i,k}^\tau \}, \quad (9.127)$$

$$\Delta p_{u,k}^B = \sum_{r=1}^N \varepsilon_{k,r}^I \{ p_{u,k}^B \}, \quad u = 2, \dots, P^B, \quad (9.128)$$

$$\Delta \bar{y}^r = \Delta p_{1,r}^B = \varepsilon^{\text{def}} \{ \bar{y}^r \} + \sum_{k=1}^N \varepsilon_{k,r}^I \{ \bar{y}^r \} + \sum_{k=1}^N \varepsilon_{r,k}^I \{ p_{1,r}^B \}, \quad (9.129)$$

$$\Delta w_k^{\text{agr}} = \sum_{r=1}^N \varepsilon_r^{\text{agr}} \{ w_k^{\text{agr}} \}. \quad (9.130)$$

Błędy propagowane przez poszczególne warstwy systemu wyznacza się następująco (rys. 9.11):

$$\varepsilon_k^\tau = \sum_{r=1}^N \varepsilon_{k,r}^I \{ \tau_k(\bar{x}) \}, \quad (9.131)$$

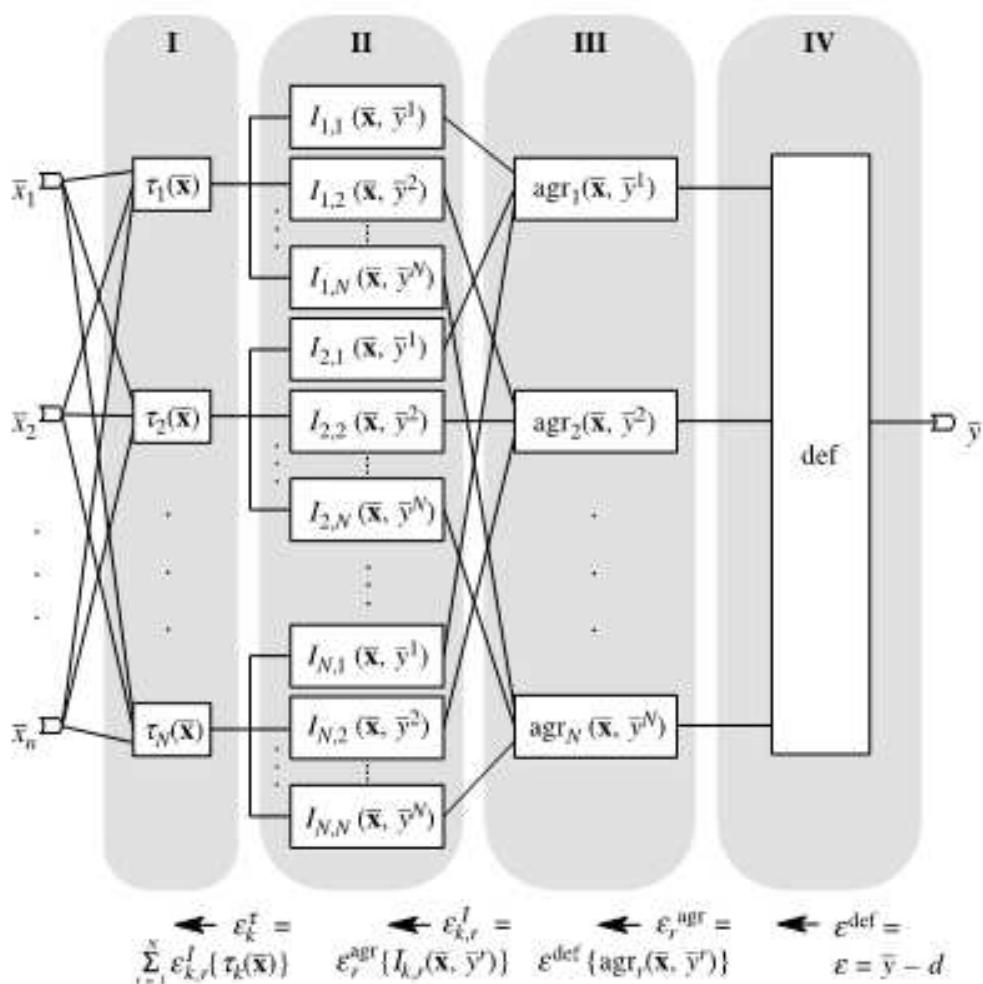
$$\varepsilon_{k,r}^I = \varepsilon_r^{\text{agr}} \{ I_{k,r}(\bar{x}, \bar{y}^r) \}, \quad (9.132)$$

$$\varepsilon_r^{\text{agr}} = \varepsilon^{\text{def}} \{ \text{agr}_r(\bar{x}, \bar{y}^r) \}, \quad (9.133)$$

$$\varepsilon^{\text{def}} = \varepsilon = \bar{y} - d. \quad (9.134)$$

Błędy propagowane przez bloki poziomów aktywności reguł systemu określa się analogicznie jak w systemie Takagi–Sugeno. Sposób wyznaczania błędów propagowanych przez bloki implikacji systemu zależy od przyjętego modelu wnioskowania (Mamdaniego lub logicznego) oraz typu zastosowanej rozmytej implikacji (S , R , Q -implikacja) w przypadku wnioskowania typu logicznego. Błędy propagowane przez bloki implikacji systemu z wnioskowaniem typu Mamdaniego określa się następująco (rys. 9.12):

$$\varepsilon_{k,r}^I \{ p_{u,k}^B \} = \varepsilon_{k,r}^I \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}, \quad (9.135)$$

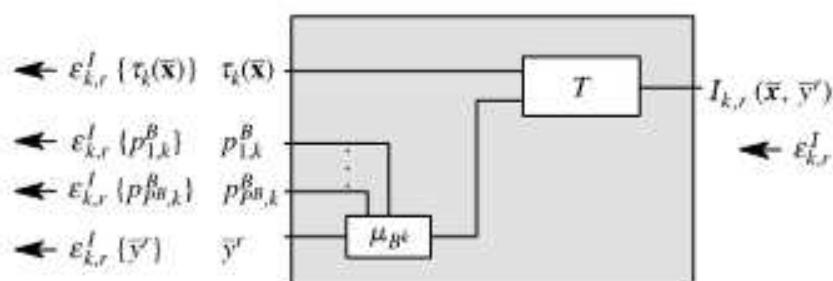


Rys. 9.11. Przepływ błędów w systemie neuronowo-rozmytym

$$\epsilon_{k,r}^I \{ \bar{y}^r \} = \epsilon_{k,r}^I \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}, \quad (9.136)$$

$$\epsilon_{k,r}^I \{ \tau_k(\bar{x}) \} = \epsilon_{k,r}^I \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \tau_k(\bar{x})}, \quad (9.137)$$

przy czym pochodne $\frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{\alpha,k}^B}$, $\frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}$, $\frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \tau_k(\bar{x})}$ oraz $\frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)}$ wyznacza się, korzystając z zależności podanych przy okazji opisu sposobu uczenia systemu Takagi–Sugeno.



Rys. 9.12. Blok implikacji systemu z wnioskowaniem typu Mamdaniego

Błędy propagowane przez bloki implikacji systemu z wnioskowaniem typu logicznego wykorzystującym *S*-implikację określa się następująco (rys. 9.13):

$$\varepsilon_{k,r}^I \{ p_{u,k}^B \} = \varepsilon_{k,r}^I \frac{\partial S \{ N(\tau_k(\bar{x})), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}, \quad (9.138)$$

$$\varepsilon_{k,r}^I \{ \bar{y}^r \} = \varepsilon_{k,r}^I \frac{\partial S \{ N(\tau_k(\bar{x})), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}, \quad (9.139)$$

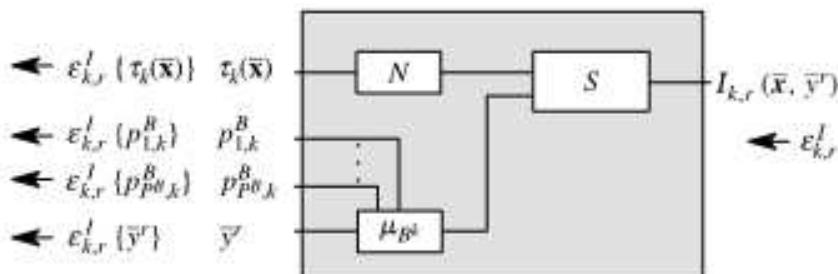
$$\varepsilon_{k,r}^I \{ \tau_k(\bar{x}) \} = \varepsilon_{k,r}^I \frac{\partial S \{ N(\tau_k(\bar{x})), \mu_{B^k}(\bar{y}^r) \}}{\partial N(\tau_k(\bar{x}))} \frac{\partial N(\tau_k(\bar{x}))}{\partial \tau_k(\bar{x})}, \quad (9.140)$$

przy czym

$$N(a) = 1 - a, \quad (9.141)$$

$$\frac{\partial N(a)}{\partial a} = -1 \quad (9.142)$$

natomiast pochodne $\frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}$ i $\frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}$ wyznacza się, korzystając z zależności podanych przy okazji opisu sposobu uczenia systemu Takagi–Sugeno.



Rys. 9.13. Blok implikacji systemu neuronowo-rozmytego z wnioskowaniem typu logicznego (S-implikacja)

Sposób wyznaczenia pochodnych cząstkowych we wzorach (9.138)–(9.140) zostanie pokazany w przykładzie 9.3. Przykład ten dotyczy bardziej ogólnego przypadku, uwzględniającego dowolną liczbę argumentów oraz ich wagę w definicji t -konormy.

Przykład 9.3

Wprowadzając zapis funkcji zakresowej do definicji ważonej t -konormy, otrzymujemy

$$S^* \left\{ \begin{array}{l} a_1, a_2, \dots, a_n; \\ w_1, w_2, \dots, w_n \end{array} \right\} = S^* \{ \mathbf{a}; \mathbf{w} \} = \sum_{i=1}^n \{ f_z(w_i) a_i \}. \quad (9.143)$$

W przypadku t -konormy algebraicznej mamy

$$S^* \{ \mathbf{a}; \mathbf{w} \} = 1 - \prod_{i=1}^n (1 - f_z(w_i) a_i). \quad (9.144)$$

Wówczas

$$\frac{\partial S^* \{ \mathbf{a}; \mathbf{w} \}}{\partial a_i} = f_z(w_i) \prod_{\substack{u=1 \\ u \neq i}}^n (1 - f_z(w_u) a_u) \quad (9.145)$$

oraz

$$\frac{\partial S^* \{ \mathbf{a}; \mathbf{w} \}}{\partial w_i} = a_i \frac{\partial f_z(w_i)}{\partial w_i} \prod_{\substack{u=1 \\ u \neq i}}^n (1 - f_z(w_u) a_u). \quad (9.146)$$

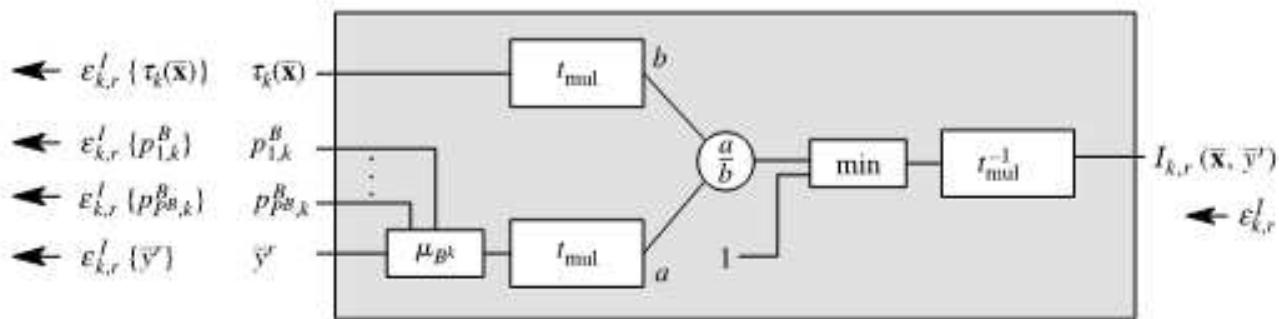
Błędy propagowane przez bloki implikacji systemu z wnioskowaniem typu logicznego wykorzystującym R -implikację określają się następująco (rys. 9.14):

$$\varepsilon_{k,r}^I \{ p_{u,k}^B \} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial t_{\text{mul}}^{-1} \left(\min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\} \right)}{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}} \frac{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}}{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}} \times \\ \times \frac{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}}{\partial t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))} \frac{\partial t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B} \end{array} \right), \quad (9.147)$$

$$\varepsilon_{k,r}^I \{ \bar{y}^r \} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial t_{\text{mul}}^{-1} \left(\min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\} \right)}{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}} \frac{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}}{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}} \times \\ \times \frac{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}}{\partial t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))} \frac{\partial t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r} \end{array} \right), \quad (9.148)$$

$$\varepsilon_{k,r}^I \{ \tau_k(\bar{x}) \} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial t_{\text{mul}}^{-1} \left(\min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\} \right)}{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}} \frac{\partial \min \left\{ 1, \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))} \right\}}{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}} \times \\ \times \frac{\partial \frac{t_{\text{mul}}(\mu_{B^k}(\bar{y}^r))}{t_{\text{mul}}(\tau_k(\bar{x}))}}{\partial t_{\text{mul}}(\tau_k(\bar{x}))} \frac{\partial t_{\text{mul}}(\tau_k(\bar{x}))}{\partial \tau_k(\bar{x})} \end{array} \right), \quad (9.149)$$

W powyższych wzorach występują pochodne operatora dzielenia oraz operatora minimum. Sposób ich wyznaczania podano w podrozdziale 10.6 następnego rozdziału.



Rys. 9.14. Blok implikacji systemu neuronowo-rozmytego z wnioskowaniem typu logicznego (R -implikacja)

Przykład 9.4

Do wygenerowania R -implikacji Goguena można zastosować multiplikatywny generator t -normy postaci

$$t_{\text{mul}}(a) = a^p, \quad p > 0. \quad (9.150)$$

Wówczas we wzorach (9.147)–(9.149) korzysta się z następujących zależności:

$$\frac{\partial t_{\text{mul}}(a)}{\partial a} = pa^{p-1}, \quad (9.151)$$

$$t_{\text{mul}}^{-1}(a) = a^{\frac{1}{p}}, \quad (9.152)$$

$$\frac{\partial t_{\text{mul}}^{-1}(a)}{\partial a} = \frac{1}{p} a^{\frac{1}{p}-1}. \quad (9.153)$$

Błędy propagowane przez bloki implikacji systemu z wnioskowaniem typu logicznego wykorzystującym Q -implikację określa się następująco (rys. 9.15):

$$\varepsilon_{k,r}^I \{ p_{u,k}^B \} = \varepsilon_{k,r}^I \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}} \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}, \quad (9.154)$$

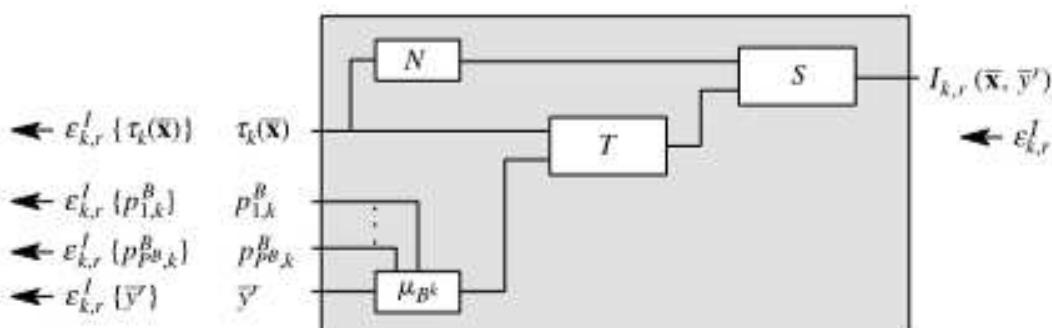
$$\varepsilon_{k,r}^I \{ \bar{y}^r \} = \varepsilon_{k,r}^I \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}} \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)} \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}, \quad (9.155)$$

$$\varepsilon_{k,r}^I \{ \tau_k(\bar{x}) \} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial N(\tau_k(\bar{x}))} \frac{\partial N(\tau_k(\bar{x}))}{\partial \tau_k(\bar{x})} + \\ + \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}} \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \tau_k(\bar{x})} \end{array} \right), \quad (9.156)$$

przy czym pochodne

$$\begin{aligned} & \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}, \quad \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}, \quad \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \tau_k(\bar{x})}, \\ & \frac{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}}{\partial \mu_{B^k}(\bar{y}^r)}, \quad \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial N(\tau_k(\bar{x}))}, \\ & \frac{\partial S \{ N(\tau_k(\bar{x})), T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \} \}}{\partial T \{ \tau_k(\bar{x}), \mu_{B^k}(\bar{y}^r) \}} \quad \text{oraz} \quad \frac{\partial N(\tau_k(\bar{x}))}{\partial \tau_k(\bar{x})} \end{aligned}$$

wyznacza się, korzystając z zależności podanych wcześniej oraz przy okazji opisu sposobu uczenia systemu Takagi–Sugeno.



Rys. 9.15. Blok implikacji systemu neuronowo-rozmytego z wnioskowaniem typu logicznego (Q -implikacja)

Błędy propagowane przez bloki agregacji systemu określają się w zależności od przyjętego modelu wnioskowania. Błędy propagowane przez bloki agregacji systemu z wnioskowaniem typu Mamdaniego wyznacza się następująco (rys. 9.16):

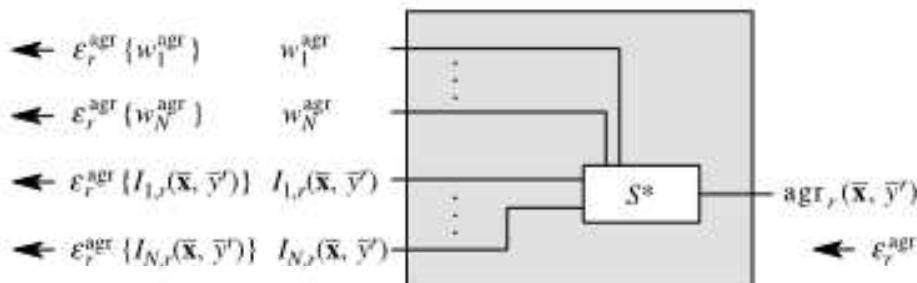
$$\varepsilon_r^{\text{agr}}\{w_k^{\text{agr}}\} = \varepsilon_r^{\text{agr}} \frac{\partial S^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial w_k^{\text{agr}}}, \quad (9.157)$$

$$\varepsilon_r^{\text{agr}}\{I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)\} = \varepsilon_r^{\text{agr}} \frac{\partial S^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}, \quad (9.158)$$

przy czym pochodne

$$\frac{\partial S^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)} \quad \text{oraz} \quad \frac{\partial S^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial w_k^{\text{agr}}}$$

wyznacza się na podstawie zależności podanych wcześniej.



Rys. 9.16. Blok agregacji systemu neuronowo-rozmytego z wnioskowaniem typu Mamdaniego

Błędy propagowane przez bloki agregacji systemu z wnioskowaniem typu logicznego określają się następująco (rys. 9.17):

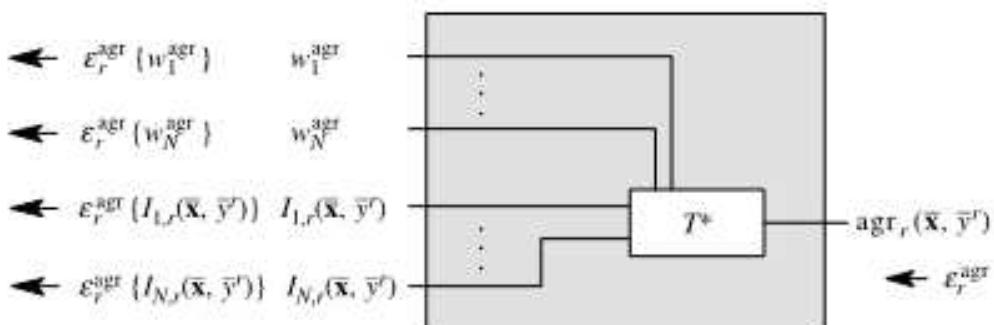
$$\varepsilon_r^{\text{agr}}\{w_k^{\text{agr}}\} = \varepsilon_r^{\text{agr}} \frac{\partial T^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial w_k^{\text{agr}}}, \quad (9.159)$$

$$\varepsilon_r^{\text{agr}}\{I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)\} = \varepsilon_r^{\text{agr}} \frac{\partial T^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}, \quad (9.160)$$

przy czym pochodne

$$\frac{\partial T^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)} \quad \text{oraz} \quad \frac{\partial T^* \left\{ \begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}} \end{array} \right\}}{\partial w_k^{\text{agr}}}$$

wyznacza się, korzystając z zależności podanych przy okazji opisu sposobu uczenia systemu Takagi–Sugeno.



Rys. 9.17. Blok agregacji systemu neuronowo-rozmytego z wnioskowaniem typu logicznego

Błędy propagowane przez blok wyostrzania systemu określają się następująco (rys. 9.18):

$$\varepsilon^{\text{def}}\{\bar{y}^r\} = \varepsilon^{\text{def}} \frac{\partial \text{def} \left(\begin{array}{c} \text{agr}_1(\bar{x}, \bar{y}^1), \dots, \text{agr}_N(\bar{x}, \bar{y}^N); \\ \bar{y}^1, \dots, \bar{y}^N \end{array} \right)}{\partial \bar{y}^r}, \quad (9.161)$$

$$\varepsilon^{\text{def}}\{\text{agr}_r(\bar{x}, \bar{y}^r)\} = \varepsilon^{\text{def}} \frac{\partial \text{def} \left(\begin{array}{c} \text{agr}_1(\bar{x}, \bar{y}^1), \dots, \text{agr}_N(\bar{x}, \bar{y}^N); \\ \bar{y}^1, \dots, \bar{y}^N \end{array} \right)}{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}, \quad (9.162)$$

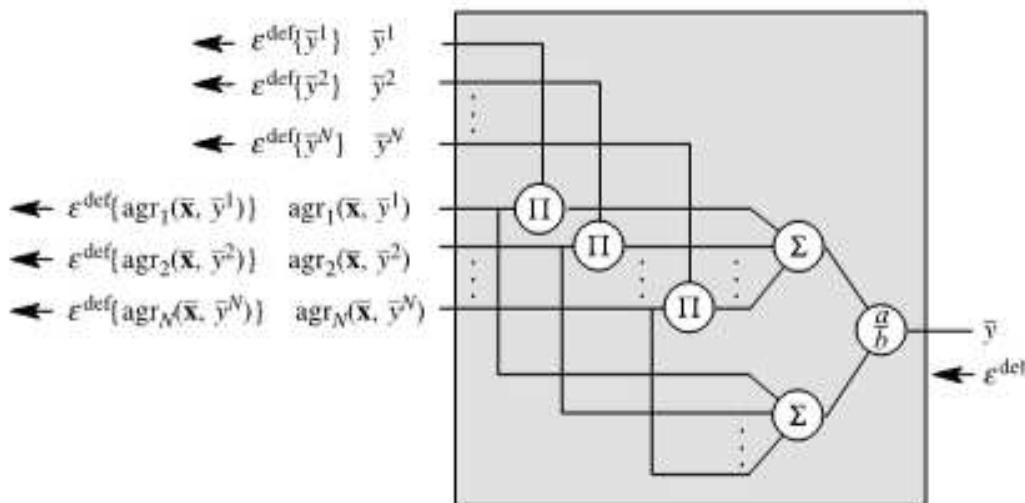
przy czym

$$\text{def}(a_1, a_2, \dots, a_n; w_1, w_2, \dots, w_n) = \text{def}(\mathbf{a}; \mathbf{w}) = \frac{\sum_{i=1}^n w_i a_i}{\sum_{i=1}^n a_i}, \quad (9.163)$$

$$\frac{\partial \text{def}(\mathbf{a}; \mathbf{w})}{\partial a_j} = (w_j - \text{def}(\mathbf{a}; \mathbf{w})) \frac{1}{\sum_{i=1}^n a_i}, \quad (9.164)$$

$$\frac{\partial \text{def}(\mathbf{a}; \mathbf{w})}{\partial w_j} = \left(a_j - \text{def}(\mathbf{a}; \mathbf{w}) \frac{\partial a_j}{\partial w_j} \right) \frac{1}{\sum_{i=1}^n a_i}. \quad (9.165)$$

Zauważmy, że zależności (9.163)–(9.165) są szczególnym przypadkiem zależności (9.113)–(9.116).



Rys. 9.18. Blok wyostrzania systemu neuronowo-rozmytego

9.7. Ocena działania systemów neuronowo-rozmytych

Rozważania prowadzone w podrozdziałach 9.3–9.5 i próba oceny badanych systemów rozmytych opierały się na wielkościach błędu jako kryterium oceny. Z rozważań tych wynika, że zwykle systemy zawierające większą liczbę uczonych parametrów pozwalały na osiągnięcie lepszych rezultatów. Jednakże pożądany system neuronowo-rozmyty powinien charakteryzować się jak najmniejszym błędem, ale jednocześnie powinien być możliwie najprostszy. Należy przypomnieć, że systemy o mniejszej liczbie uczonych parametrów charakteryzują się m.in. lepszymi zdolnościami generalizowania uzyskanych rozwiązań. W tym miejscu warto wspomnieć o tzw. *zasadzie oszczędności* [235]. Zasada ta jest bardzo użyteczna przy wyznaczaniu właściwego rzędu modelu. Można ją sformułować w następujący sposób: *spośród dwóch alternatywnych i satysfakcjonujących nas modeli wybieramy ten, w którym występuje mniej niezależnych parametrów*. Zasada ta pozostaje w zgodzie ze zdrowym rozsądkiem: „*nie wprowadzaj do opisu procesu dodatkowych parametrów, jeśli nie są one niezbędne*”.

Metody estymacji rzędu systemu najlepiej zostały rozwinięte dla procesów autoregresji [107, 132, 202]. Szereg czasowy $u(n), u(n-1), \dots, u(n-p)$ jest procesem autoregresji rzędu p , jeżeli spełnia równanie różnicowe

$$u(n) + \alpha_1 u(n-1) + \dots + \alpha_p u(n-p) = e(n) \quad (9.166)$$

lub równoważnie

$$u(n) = - \sum_{k=1}^p \alpha_k u(n-k) + e(n), \quad (9.167)$$

gdzie $\alpha_1, \dots, \alpha_p$ są współczynnikami procesu, natomiast $e(n)$ jest szumem białym

$$\mathbf{E}[e(n)] = 0, \quad \mathbf{E}[e(n)e(m)] = \begin{cases} \sigma^2 & \text{dla } n = m, \\ 0 & \text{dla } n \neq m. \end{cases} \quad (9.168)$$

W teorii autoregresji stosuje się kryteria pozwalające estymować rzad predyktora p , wyznaczając błąd predykcji \hat{Q}_p na podstawie ciągu uczącego o długości M . Najważniejsze z nich to kryterium informacyjne Akaike (AIC), metoda Schwarza oraz metoda końcowego błędu predykcji (FPE).

W kolejnym punkcie najpierw przedstawimy podstawowe kryteria oceny modeli (z uwzględnieniem ich złożoności), pierwotnie stosowane do estymacji rzędu procesów autoregresji, a następnie zostaną one zaadaptowane do oceny efektywności systemów neuronowo-rozmytych. Przez *efektywność działania systemu neuronowo-rozmytego* rozumiemy osiągniętą dokładność działania takiego systemu (wyrażoną błędem średnim kwadratowym lub liczbą błędnie sklasyfikowanych próbek) w kontekście jego rozmiaru. Przez *rozmiar systemu* rozumiemy liczbę wszystkich parametrów, które podlegają uczeniu. Przedstawimy również koncepcję tzw. *linii izokryterialnych*, które pozwalają rozwiązać problem kompromisu między błędem działania systemu, a liczbą parametrów opisujących ten system.

9.7.1. Kryteria oceny modeli z uwzględnieniem ich złożoności

Poniżej przedstawiono dwa ogólne kryteria uwzględniające złożoność modelu, wskazano na zależności między tymi kryteriami, a następnie pokazano ich szczególne postacie.

9.7.1.1. Kryterium A

Ogólna postać kryterium A uwzględniającego złożoność modelu jest dana wzorem

$$W(p) = \widehat{Q}_p [1 + \beta(M, p)], \quad (9.169)$$

w którym \widehat{Q}_p jest błędem średnim kwadratowym, natomiast $\beta(M, p)$ jest funkcją długości ciągu uczącego M i liczby parametrów p modelu. Aby eliminować zbyt złożone struktury (zgodnie z zasadą oszczędności) zakładamy, że

$$\lim_{p \rightarrow \infty} \beta(M, p) = \infty. \quad (9.170)$$

Jednocześnie, aby obecność członu karzącego w wyrażeniu (9.169) nie utrudniała zaobserwowania zmniejszania się wartości błędu średniego kwadratowego \widehat{Q}_p ze wzrostem złożoności modelu, przyjmujemy, że

$$\lim_{M \rightarrow \infty} \beta(M, p) = 0. \quad (9.171)$$

Typowy wybór to $\beta(M, p) = 2p/M$ i wówczas

$$W(p) = \widehat{Q}_p \left[1 + \frac{2p}{M} \right]. \quad (9.172)$$

9.7.1.2. Kryterium B

Alternatywnym kryterium w stosunku do wzoru (9.169) może być następująca formuła:

$$W(p) = M \log \widehat{Q}_p + \gamma(M, p), \quad (9.173)$$

w której człon dodatkowy $\gamma(M, p)$ powinien uwzględniać karę za przyjęcie modeli zbyt wysokiego rzędu. Łatwo sprawdzić, że jeśli

$$\gamma(M, p) = M\beta(M, p), \quad (9.174)$$

to kryteria (9.169) i (9.173) są asymptotycznie równoważne.

Poniżej zostaną przedstawione podstawowe metody kompromisowego doboru rzędu modelu. Większość tych metod to przypadki szczególnie wyżej przedstawionego kryterium A lub B.

9.7.1.3. Metoda kryterium informacyjnego Akaike (AIC)

Przyjęcie $\gamma(M, p) = 2p$ w kryterium (9.173) daje tzw. *informacyjne kryterium Akaike* (ang. *Akaike Information Criterion*). Złożoność systemu p można znaleźć, szukając najmniejszej wartości wyrażenia

$$\text{AIC}(p) = M \ln \widehat{Q}_p + 2p. \quad (9.175)$$

9.7.1.4. Metoda końcowego błędu predykcji (FPE)

Kryterium FPE również zaproponował Akaike. W *metodzie końcowego błędu predykcji* (ang. *Final Prediction Error*), która nie wynika z ogólnych formuł (9.169) i (9.173), złożoność systemu p znajdziemy, szukając najmniejszej wartości wyrażenia

$$\text{FPE}(p) = \frac{M + p}{M - p} \widehat{Q}_p. \quad (9.176)$$

W wyrażeniu (9.175) wraz ze wzrostem parametru p czynnik $\frac{M+p}{M-p}$ rośnie, natomiast wartość błędu średniego kwadratowego \widehat{Q}_p maleje. Zauważmy, że dla dużych wartości M można stosować następujące przybliżenie:

$$\text{FPE}(p) = \widehat{Q}_p \left[1 + \frac{2p/M}{1 - p/M} \right] \approx \widehat{Q}_p \left[1 + \frac{2p}{M} \right], \quad (9.177)$$

które jest typu (9.169), tzn.

$$\beta(M, p) = \frac{2p}{M}. \quad (9.178)$$

Wyrażenia (9.169) i (9.173) są asymptotycznie równoważne, jeśli spełniony jest warunek (9.174), a zatem

$$\gamma(M, p) = 2p. \quad (9.179)$$

W konsekwencji

$$\text{FPE}(p) \approx \text{AIC}(p) = M \ln \widehat{Q}_p + 2p. \quad (9.180)$$

Kryteria FPE oraz AIC wykazują tendencję do wyboru modelu zbyt małego rzędu. Dlatego w literaturze [235] proponuje się trzy inne metody opisane poniżej.

9.7.1.5. Metoda Schwarza

Przyjęcie $\gamma(M, p) = p \log M$ w kryterium (9.173) daje tzw. *kryterium Schwarza*. W metodzie tej złożoność systemu p można znaleźć, szukając najmniejszej wartości wyrażenia

$$S(p) = M \ln \widehat{Q}_p + p \ln M. \quad (9.181)$$

9.7.1.6. Metoda Söderströma i Stoicy

Przyjęcie $\gamma(M, p) = 2pc \log(\log M)$, gdzie $c \geq 1$, w kryterium (9.173) daje tzw. *kryterium Söderströma i Stoicy*. W metodzie tej złożoność systemu p znajduje się, szukając najmniejszej wartości wyrażenia

$$H(p) = M \ln \widehat{Q}_p + 2pc \log(\log M). \quad (9.182)$$

9.7.1.7. Metoda CAT

W metodzie CAT (ang. *Criterion Autoregressive Transfer Function*) złożoność systemu p można znaleźć, szukając najmniejszej wartości wyrażenia

$$\text{CAT}(p) = \frac{1}{M} \sum_{i=1}^p \frac{1}{\overline{Q}_i} - \frac{1}{\overline{Q}_p}, \quad (9.183)$$

w którym $\overline{Q}_i = \frac{m}{M-i} \widehat{Q}_i$.

Opisane powyżej metody wyznaczania rzędu modelu pierwotnie zostały zaproponowane do analizy procesów autoregresji danych z użyciem wzoru (9.166). Jednakże należy stwierdzić, że metody te pozwalają wyznaczyć właściwy rząd modelu niezależnie od tego, czy system należy do klasy struktur modelu, czy też nie [235].

9.7.2. Metoda linii izokryterialnych

Opisane w poprzednim punkcie metody estymacji rzędu predykcji zaadaptujemy teraz do oceny systemów rozmytych. Dzięki temu poszukiwanie pożdanego systemu rozmytego na podstawie dwóch kryteriów (liczby parametrów oraz błędu średniego kwadratowego) sprowadzimy do jednego, wybranego kryterium, tzn. AIC, Schwarza lub FPE. Zostały one zaadaptowane na potrzeby oceny systemów neuronowo-rozmytych w następującej formie:

$$\text{AIC}(p, \widehat{Q}_p) = M \ln \widehat{Q}_p + 2p, \quad (9.184)$$

$$S(p, \widehat{Q}_p) = M \ln \widehat{Q}_p + p \ln M, \quad (9.185)$$

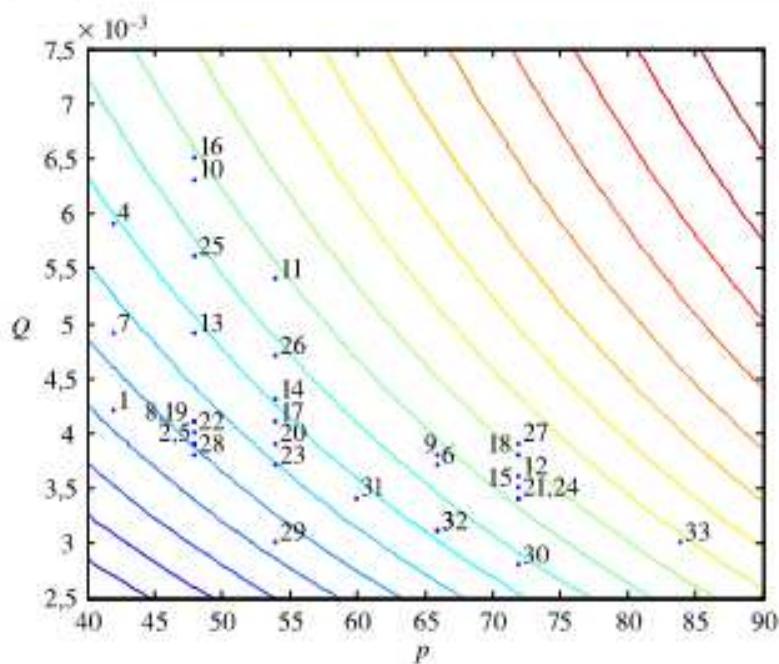
$$\text{FPE}(p, \widehat{Q}_p) = \frac{Mn + p}{Mn - p} \widehat{Q}_p, \quad (9.186)$$

gdzie p jest liczbą parametrów systemu podlegających uczeniu (liczba parametrów wszystkich funkcji przynależności oraz liczba wszystkich wag, jeśli występują w danym systemie), \widehat{Q}_p jest miarą błędu wykorzystywaną w symulacjach opisanych w podrozdziałach 9.3–9.5, M określa liczbę próbek w ciągu uczącym, natomiast n jest liczbą wejść systemu. Iloczyn $M \cdot n$ może być zatem traktowany jako miara wielkości rozwiązywanego zagadnienia. Tabele 9.62 i 9.63 zawierają wyliczone wartości kryteriów dla poszczególnych badanych struktur w przypadku ciągów uczącego i testowego użytych w zagadnieniu polimeryzacji. Na rysunkach 9.19–9.24 zamieszczono punkty odpowiadające poszczególnym badanym systemom neuronowo-rozmytym. Współrzędna p określa liczbę parametrów danego systemu, współrzędna Q określa błąd, z jakim system realizuje postawione przed nim zadanie. Linie izokryterialne przedstawiają stałe wartości kryterium AIC, Schwarza i FPE, przy różnych wartościach błędów i liczby parametrów. Takie podejście pozwala rozwiązać problem kompromisu między błędem działania systemu a liczbą parametrów opisujących ten system. Punkty leżące na liniach izokryterialnych o tych samych wartościach kryterium AIC, Schwarza lub FPE charakteryzują systemy neuronowo-rozmyte tworzące zbiór Pareto. W zbiorze Pareto nie można poprawić żadnego z dwóch sprzecznych kryteriów (błąd średni kwadratowy *versus* rozmiar systemu), nie pogarszając pozostałego. Punkty leżące na liniach izokryterialnych o najmniejszych wartościach kryterium AIC, Schwarza lub FPE charakteryzują systemy neuronowo-rozmyte, które nazwano suboptimalnymi. Suboptimalne systemy neuronowo-rozmyte przedstawione na wykresach zapewniają najmniejszą wartość kryteriów statystycznych w ramach przebadanych struktur (nie stosuje się terminologii „optimalne systemy”, gdyż nie przebadano wszystkich możliwych struktur).

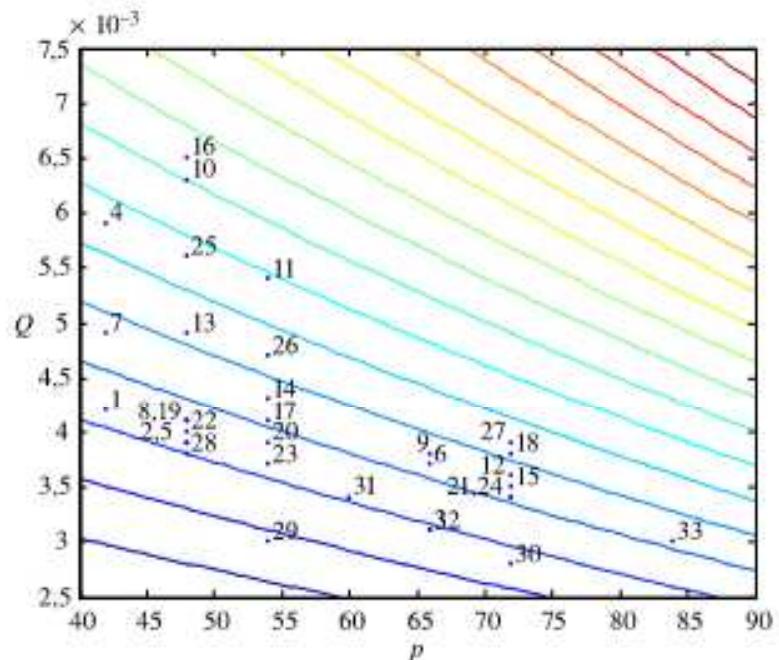
Z tabel 9.62 i 9.63 oraz z rysunków wynika, że zarówno dla ciągu uczącego, jak i testowego kryterium AIC najlepiej ocenia system 1 (uproszczona struktura Larsena), a zaraz za nim system 29 (struktura Zadeha z wagami reguł), kryterium FPE — system 29, kryterium Schwarza — zdecydowanie system 1.

Tabela 9.62. Wartości kryteriów dla ciągu uczącego

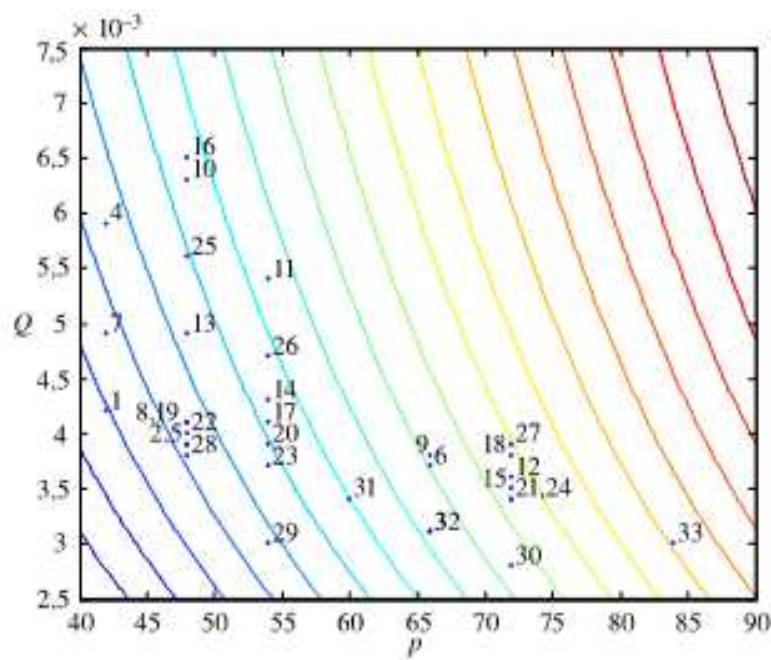
Lp.	Struktura	Polimeryzacja				
		błąd	p	AIC	FPE	Schwarz
1	Larsena uproszczona	0,0042	42	-299,09	0,0063	-204,65
2	Larsena uproszczona z wagami reguł	0,0039	48	-292,27	0,0062	-184,35
3	Larsena uproszczona z wagami wejść i reguł	0,0031	66	-272,34	0,0059	-123,94
4	Łukasiewicza uproszczona	0,0059	42	-275,30	0,0089	-180,86
5	Łukasiewicza uproszczona z wagami reguł	0,0039	48	-292,27	0,0062	-184,35
6	Łukasiewicza uproszczona z wagami wejść i reguł	0,0037	66	-259,96	0,0071	-111,56
7	Zadeha uproszczona	0,0049	42	-288,30	0,0074	-193,86
8	Zadeha uproszczona z wagami reguł	0,0041	48	-288,77	0,0065	-180,85
9	Zadeha uproszczona z wagami wejść i reguł	0,0038	66	-258,09	0,0073	-109,69
10	Binarna	0,0063	48	-258,70	0,01	-150,78
11	Binarna z wagami reguł	0,0054	54	-257,49	0,0091	-136,08
12	Binarna z wagami wejść i reguł	0,0036	72	-249,88	0,0074	-87,99
13	Larsena	0,0049	48	-276,30	0,0078	-168,37
14	Larsena z wagami reguł	0,0043	54	-273,44	0,0073	-152,02
15	Larsena z wagami wejść i reguł	0,0035	72	-251,85	0,0072	-89,96
16	Łukasiewicza	0,0065	48	-256,52	0,0104	-148,59
17	Łukasiewicza z wagami reguł	0,0041	54	-276,77	0,0069	-155,36
18	Łukasiewicza z wagami wejść i reguł	0,0038	72	-246,09	0,0078	-84,20
19	Mamdaniego	0,0041	48	-288,77	0,0065	-180,85
20	Mamdaniego z wagami reguł	0,0039	54	-280,27	0,0066	-158,86
21	Mamdaniego z wagami wejść i reguł	0,0034	72	-253,88	0,0069	-91,99
22	Reichenbacha	0,0040	48	-290,50	0,0064	-182,57
23	Reichenbacha z wagami reguł	0,0037	54	-283,96	0,0063	-162,54
24	Reichenbacha z wagami wejść i reguł	0,0034	72	-253,88	0,0069	-91,990
25	Willmotta	0,0056	48	-266,95	0,0089	-159,02
26	Willmotta z wagami reguł	0,0047	54	-267,21	0,008	-145,79
27	Willmotta z wagami wejść i reguł	0,0039	72	-244,27	0,008	-82,38
28	Zadeha	0,0038	48	-294,09	0,0061	-186,17
29	Zadeha z wagami reguł	0,0030	54	-298,64	0,0051	-177,22
30	Zadeha z wagami wejść i reguł	0,0028	72	-267,47	0,0057	-105,58
31	Takagi-Sugeno	0,0034	60	-277,88	0,0061	-142,97
32	Takagi-Sugeno z wagami reguł	0,0031	66	-272,34	0,0059	-123,94
33	Takagi-Sugeno z wagami wejść i reguł	0,0030	84	-238,64	0,007	-49,77



Rys. 9.19. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium Akaike dla ciągu uczącego — zagadnienie polimeryzacji



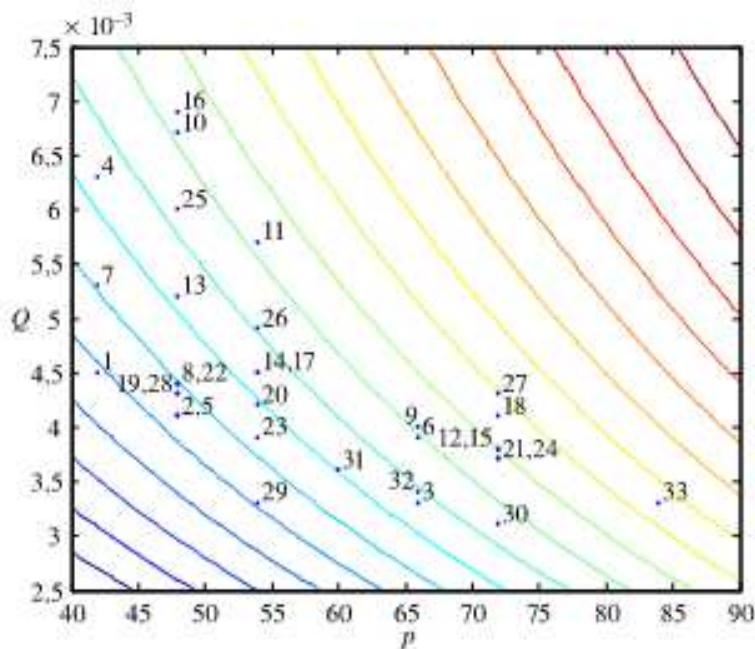
Rys. 9.20. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium FPE dla ciągu uczącego — zagadnienie polimeryzacji



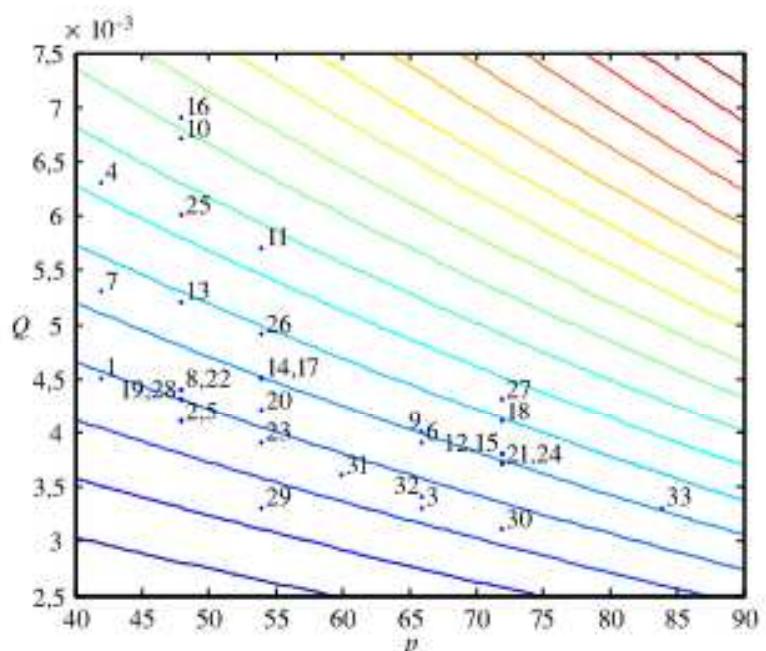
Rys. 9.21. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium Schwarza dla ciągu uczącego — zagadnienie polimeryzacji

Tabela 9.63. Wartości kryteriów dla ciągu testowego

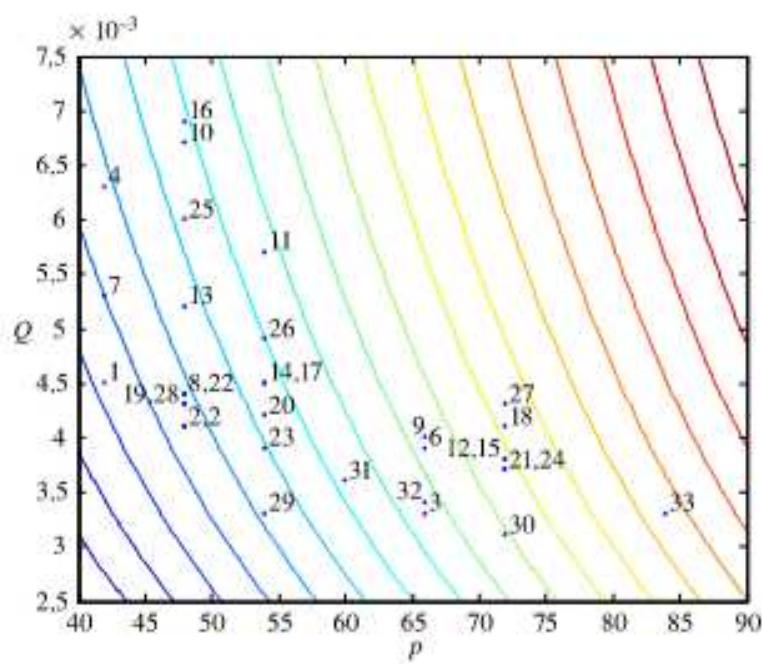
Lp.	Struktura	Polimeryzacja				
		błąd	p	AIC	FPE	Schwarz
1	Larsena uproszczona	0,0045	42	-294,26	0,0068	-199,82
2	Larsena uproszczona z wagami reguł	0,0041	48	-288,77	0,0065	-180,85
3	Larsena uproszczona z wagami wejść i reguł	0,0033	66	-267,97	0,0063	-119,57
4	Łukasiewicza uproszczona	0,0063	42	-270,70	0,0095	-176,27
5	Łukasiewicza uproszczona z wagami reguł	0,0041	48	-288,77	0,0065	-180,85
6	Łukasiewicza uproszczona z wagami wejść i reguł	0,0039	66	-256,27	0,0075	-107,87
7	Zadeha uproszczona	0,0053	42	-282,80	0,0080	-188,37
8	Zadeha uproszczona z wagami reguł	0,0044	48	-283,83	0,0070	-175,90
9	Zadeha uproszczona z wagami wejść i reguł	0,0040	66	-254,50	0,0077	-106,10
10	Binarna	0,0067	48	-254,40	0,0107	-146,47
11	Binarna z wagami reguł	0,0057	54	-253,71	0,0096	-132,29
12	Binarna z wagami wejść i reguł	0,0038	72	-246,09	0,0078	-84,20
13	Larsena	0,0052	48	-272,14	0,0083	-164,21
14	Larsena z wagami reguł	0,0045	54	-270,26	0,0076	-148,84
15	Larsena z wagami wejść i reguł	0,0038	72	-246,09	0,0078	-84,20
16	Łukasiewicza	0,0069	48	-252,34	0,0110	-144,41
17	Łukasiewicza z wagami reguł	0,0045	54	-270,26	0,0076	-148,84
18	Łukasiewicza z wagami wejść i reguł	0,0041	72	-240,77	0,0084	-78,88
19	Mamdaniego	0,0043	48	-285,44	0,0068	-177,51
20	Mamdaniego z wagami reguł	0,0042	54	-275,09	0,0071	-153,67
21	Mamdaniego z wagami wejść i reguł	0,0037	72	-247,96	0,0076	-86,07
22	Reichenbacha	0,0044	48	-283,83	0,0070	-175,90
23	Reichenbacha z wagami reguł	0,0039	54	-280,27	0,0066	-158,86
24	Reichenbacha z wagami wejść i reguł	0,0037	72	-247,96	0,0076	-86,07
25	Willmotta	0,0060	48	-262,12	0,0096	-154,19
26	Willmotta z wagami reguł	0,0049	54	-264,30	0,0083	-142,88
27	Willmotta z wagami wejść i reguł	0,0043	72	-237,44	0,0088	-75,55
28	Zadeha	0,0043	48	-285,44	0,0068	-177,51
29	Zadeha z wagami reguł	0,0033	54	-291,97	0,0056	-170,55
30	Zadeha z wagami wejść i reguł	0,0031	72	-260,34	0,0063	-98,45
31	Takagi-Sugeno	0,0036	60	-273,88	0,0065	-138,97
32	Takagi-Sugeno z wagami reguł	0,0034	66	-265,88	0,0065	-117,48
33	Takagi-Sugeno z wagami wejść i reguł	0,0033	84	-231,97	0,0077	-43,09



Rys. 9.22. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium Akaike dla ciągu testowego — zagadnienie polimeryzacji

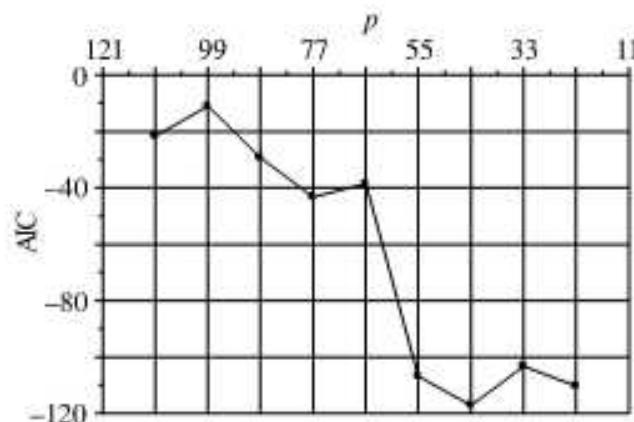


Rys. 9.23. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium FPE dla ciągu testowego — zagadnienie polimeryzacji

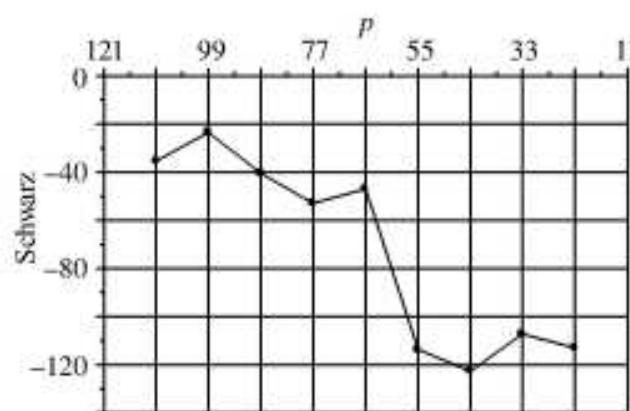


Rys. 9.24. Linie izokryterialne: wyniki osiągnięte przez poszczególne systemy dla kryterium Schwarza dla ciągu testowego — zagadnienie polimeryzacji

Analogiczne linie izokryterialne czytelnik z łatwością może wykreślić dla problemów HANG, NDP i modelowania smaku ryżu. Po wykreśleniu tych linii można sprawdzić, że w przypadku zagadnienia HANG, zarówno dla ciągu uczącego, jak i testowego, kryteria AIC i FPE wskazują na system 23 (struktura Reichenbacha z wagami reguł), natomiast kryterium Schwarza — na system 1 (uproszczona struktura Larsena). W przypadku zagadnienia NDP wszystkie trzy kryteria, zarówno dla ciągu uczącego, jak i testowego, wskazują wybór systemu 3 (uproszczona struktura Larsena z wagami wejść i reguł). W przypadku modelowania smaku ryżu, zarówno dla ciągu uczącego, jak i testowego, kryteria AIC oraz Schwarza wskazują, że najlepszy jest system 1 (uproszczona struktura Larsena), natomiast według kryterium FPE jest to system 20 (struktura Mamdaniego z wagami reguł).



Rys. 9.25. Wartości kryterium Akaiki



Rys. 9.26. Wartości kryterium Schwarza

We wszystkich symulacjach dotyczących przeprowadzonych badano efektywność działania systemów neuronowo-rozmytych, przyjmując ustaloną liczbę reguł w celu rozwiązania konkretnego zadania. Dzięki temu możliwe było porównanie 33 różnych systemów neuronowo-rozmytych. Należy podkreślić, że metodę linii izokryterialnych można również zastosować do odpowiedniego zaprojektowania każdego z tych systemów. Koncentrując się na jednym konkretnym systemie neuronowo-rozmytym, można dobrać liczbę reguł, która zapewni najmniejszą wartość jednego z kryteriów wymienionych w punkcie 9.7.1. Na przykład do problemu modelowania smaku ryżu zastosowano uproszczoną strukturę Larsena, zmieniając stopniowo liczbę reguł z 10 do 2. Poszczególne systemy charakteryzują się następującą liczbą parametrów: 110, 99, 88, 77, 66, 55, 44, 33 i 22. Na rysunkach 9.25 i 9.26 pokazano przebieg zależności kryterium AIC oraz Schwarza od liczby parametrów. Jak wynika z wykresów, kryteria AIC i Schwarza sugerują przyjęcie czterech reguł. W przeprowadzonych symulacjach problem modelowania smaku ryżu analizowano, przyjmując 5 reguł, co można wiązać z przyjęciem zasady ostrożności.

9.8. Uwagi

W rozdziale tym przedmiotem badań były systemy neuronowo-rozmyte typu Mamdaniego, logicznego i Takagi–Sugeno. Z przeprowadzonych symulacji wynika, że uwzględnienie wag odzwierciedlających ważność reguł oraz ważność zmiennych lingwistycznych

w poprzednikach reguł znacznie poprawia działanie systemów neuronowo-rozmytych. Systemy Takagi–Sugeno charakteryzują się najmniejszym błędem średnim kwadratowym, ale rezultat ten uzyskuje się przy dużej liczbie parametrów. Struktury rozszerzone (charakteryzujące się obszerniejszą informacją o funkcjach przynależności zbiorów rozmytych występujących w następnikach reguł) dają lepsze rezultaty niż struktury uproszczone. Ponadto w rozdziale tym przedstawiono zagadnienie osiągnięcia kompromisu pomiędzy błędem działania systemu a liczbą opisujących go parametrów. Z analizy linii izokryterialnych odpowiadających poszczególnym symulacjom wynika, że najczęściej systemem najlepszym w sensie przyjętych kryteriów jest uproszczona struktura Larsena dana wzorem (9.25). Systemy typu logicznego badali w swoich monografiach Czogała i Łęski [34], Rutkowska [187] oraz Rutkowski [225]. Różne podejścia do zagadnienia projektowania sieci neuronowo-rozmytych podano w pracach [65, 126, 142, 145, 148, 149, 176, 185, 186, 213, 214, 216, 239, 253, 254]. Struktury neuronowo-rozmyte w połączeniu z teorią zbiorów przybliżonych zaproponował Nowicki [151, 152], natomiast w połączeniu z teorią zbiorów rozmytych typu 2 przedstawił Starczewski [238]. Relacyjne systemy neuronowo-rozmyte analizował Scherer [231]. Metodę uczenia struktur neuronowo-rozmytych opracował Piliński [172–174]. Kryteria oceny modeli z uwzględnieniem ich złożoności szczegółowo omówiono w monografii [235].

Elastyczne systemy neuronowo-rozmyte

10.1. Wprowadzenie

W poprzednim rozdziale rozważaliśmy systemy neuronowo-rozmyte typu Mamdaniego oraz typu logicznego. W tym rozdziale skonstruujemy system neuronowo-rozmyty, którego sposób wnioskowania (Mamdaniego lub logiczny) zostanie znaleziony w wyniku procesu uczenia. Struktura takiego systemu będzie się zmieniać w trakcie uczenia. Jego realizacja będzie możliwa dzięki specjalnie skonstruowanym przełączanym normom trójkątnym. Przełączane normy trójkątne, zastosowane do agregacji poszczególnych reguł, po zakończeniu procesu uczenia przybiorą formę klasycznej t -normy lub t -konormy. W sposób analogiczny zostaną skonstruowane przełączane implikacje, które po zakończeniu procesu uczenia przybiorą formę „implikacji inżynierskiej” lub rozmytej S -implikacji. Ponadto, do konstrukcji systemów neuronowo-rozmytych wprowadzimy koncepcję miękkich norm trójkątnych, parametryzowanych norm trójkątnych oraz już wcześniej stosowanych w tym rozdziale wag opisujących ważność poszczególnych reguł oraz przesłanek w tych regułach.

10.2. Miękkie normy trójkątne

Miękkie odpowiedniki norm trójkątnych definiujemy następująco:

$$\tilde{T}\{\mathbf{a}; \alpha\} = (1 - \alpha) \frac{1}{n} \sum_{i=1}^n a_i + \alpha T\{a_1, \dots, a_n\} \quad (10.1)$$

oraz

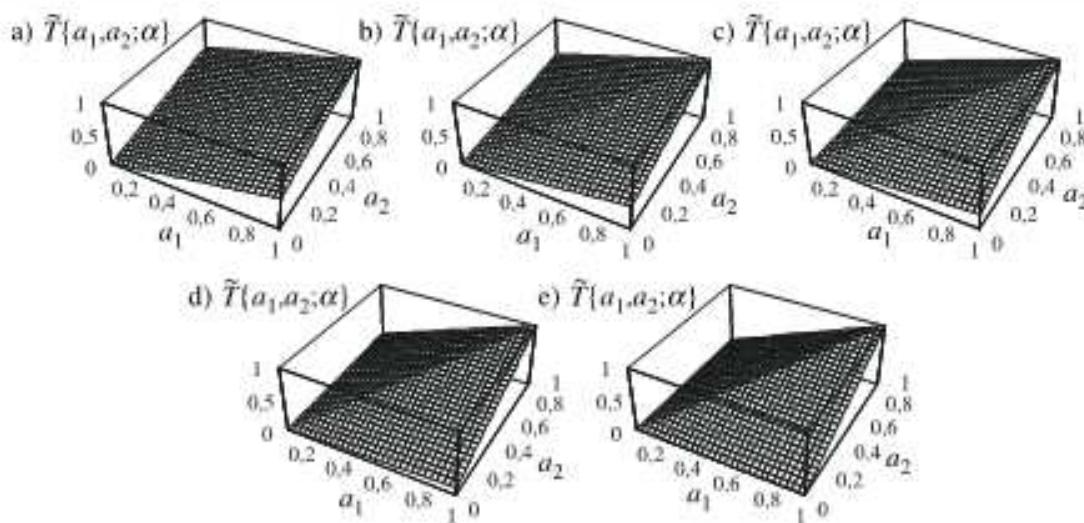
$$\tilde{S}\{\mathbf{a}; \alpha\} = (1 - \alpha) \frac{1}{n} \sum_{i=1}^n a_i + \alpha S\{a_1, \dots, a_n\}, \quad (10.2)$$

przy czym $\mathbf{a} = [a_1, \dots, a_n]$ oraz $\alpha \in [0, 1]$. Powyższe operatory pozwalają na płynne balansowanie między średnią arytmetyczną argumentów a_1, \dots, a_n a klasycznym operatorem t -normy lub t -konormy.

Przykład 10.1

Miękką t -normę Zadeha (typu min) definiujemy następująco:

$$\tilde{T}\{a_1, a_2; \alpha\} = (1 - \alpha) \frac{1}{2}(a_1 + a_2) + \alpha \min\{a_1, a_2\}. \quad (10.3)$$



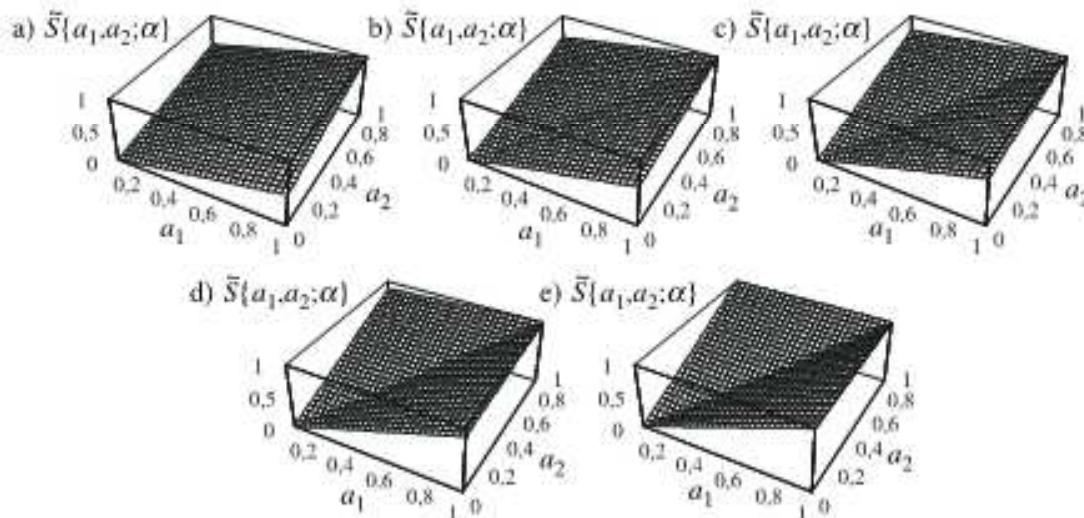
Rys. 10.1. Hiperpłaszczyzny funkcji (10.3) dla a) $\alpha = 0,00$, b) $\alpha = 0,25$, c) $\alpha = 0,50$, d) $\alpha = 0,75$, e) $\alpha = 1,00$

Jej działanie ilustruje rysunek 10.1.

Miękka t -konorma Zadeha przybiera postać

$$\tilde{S}\{a_1, a_2; \alpha\} = (1 - \alpha)\frac{1}{2}(a_1 + a_2) + \alpha \max\{a_1, a_2\}. \quad (10.4)$$

Jej działanie ilustruje rysunek 10.2.



Rys. 10.2. Hiperpłaszczyzny funkcji (10.4) dla a) $\alpha = 0,00$, b) $\alpha = 0,25$, c) $\alpha = 0,50$, d) $\alpha = 0,75$, e) $\alpha = 1,00$

Jak pamiętamy, „implikację inżynierską” definiujemy poprzez t -normę. Miękki odpowiednik tej implikacji zapisujemy następująco:

$$\tilde{I}(a, b; \beta) = (1 - \beta)\frac{1}{2}(a + b) + \beta T\{a, b\}. \quad (10.5)$$

Miękka S -implikacja przybiera postać

$$\tilde{I}(a, b; \beta) = (1 - \beta)\frac{1}{2}(1 - a + b) + \beta S\{1 - a, b\}, \quad (10.6)$$

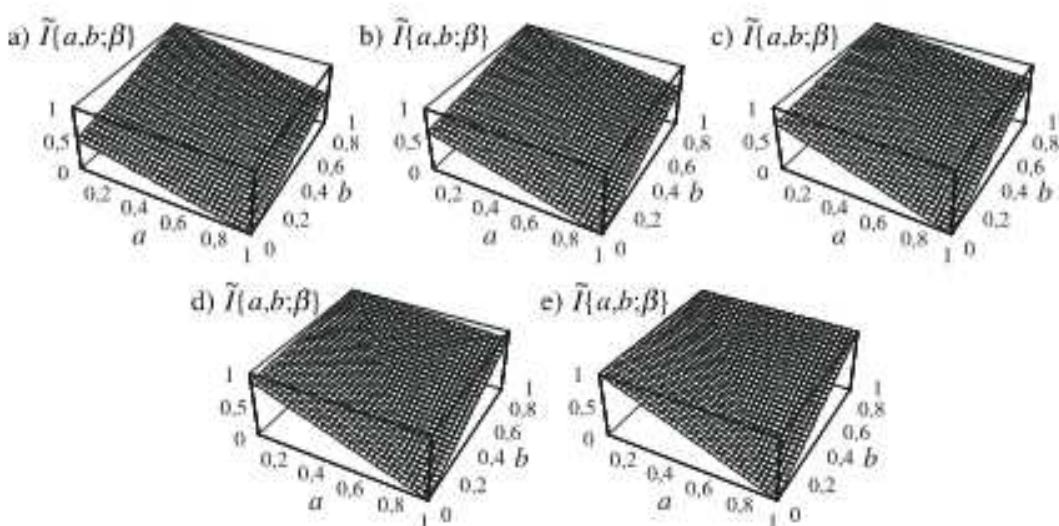
przy czym $\beta \in [0, 1]$ w obu przypadkach.

Przykład 10.2

Miękka S -implikacja binarna dana jest wzorem

$$\tilde{I}(a, b; \beta) = (1 - \beta) \frac{1}{2}(1 - a + b) + \beta \max\{1 - a, b\}. \quad (10.7)$$

Jej działanie ilustruje rysunek 10.3.



Rys. 10.3. Hiperplaszczyzny funkcji (10.7) dla a) $\beta = 0,00$, b) $\beta = 0,25$, c) $\beta = 0,50$, d) $\beta = 0,75$, e) $\beta = 1,00$

Do konstrukcji systemów typu Mamdaniego można wykorzystać następujące miękkie normy trójkątne:

- $\tilde{T}_1\{\mathbf{a}; \alpha^\tau\} = (1 - \alpha^\tau) \frac{1}{n} \sum_{i=1}^n a_i + \alpha^\tau T_{i=1}^n\{a_i\}$ do agregacji przesłanek w poszczególnych regułach;
- $\tilde{T}_2\{b_1, b_2; \alpha^I\} = (1 - \alpha^I) \frac{1}{2}(b_1 + b_2) + \alpha^I T\{b_1, b_2\}$ do połączenia przesłanek i następników reguł;
- $\tilde{S}\{\mathbf{c}; \alpha^{\text{agr}}\} = (1 - \alpha^{\text{agr}}) \frac{1}{N} \sum_{k=1}^N c_k + \alpha^{\text{agr}} S_{k=1}^N\{c_k\}$ do agregacji reguł,

gdzie n jest liczbą wejść, natomiast N jest liczbą reguł.

Do konstrukcji systemów typu logicznego wykorzystujących S -implikację można użyć następujących miękkich norm trójkątnych:

- $\tilde{T}_1\{\mathbf{a}; \alpha^\tau\} = (1 - \alpha^\tau) \frac{1}{n} \sum_{i=1}^n a_i + \alpha^\tau T_{i=1}^n\{a_i\}$ do agregacji przesłanek w poszczególnych regułach;
- $\tilde{S}\{b_1, b_2; \alpha^I\} = (1 - \alpha^I) \frac{1}{2}(1 - b_1 + b_2) + \alpha^I S\{1 - b_1, b_2\}$ do połączenia przesłanek i następników reguł;
- $\tilde{T}_2\{\mathbf{c}; \alpha^{\text{agr}}\} = (1 - \alpha^{\text{agr}}) \frac{1}{N} \sum_{k=1}^N c_k + \alpha^{\text{agr}} T_{k=1}^N\{c_k\}$ do agregacji reguł,

gdzie n jest liczbą wejść natomiast N jest liczbą reguł. Należy podkreślić, że parametry α^τ , α^I oraz α^{agr} można znaleźć w wyniku uczenia.

10.3. Parametryzowane normy trójkątne

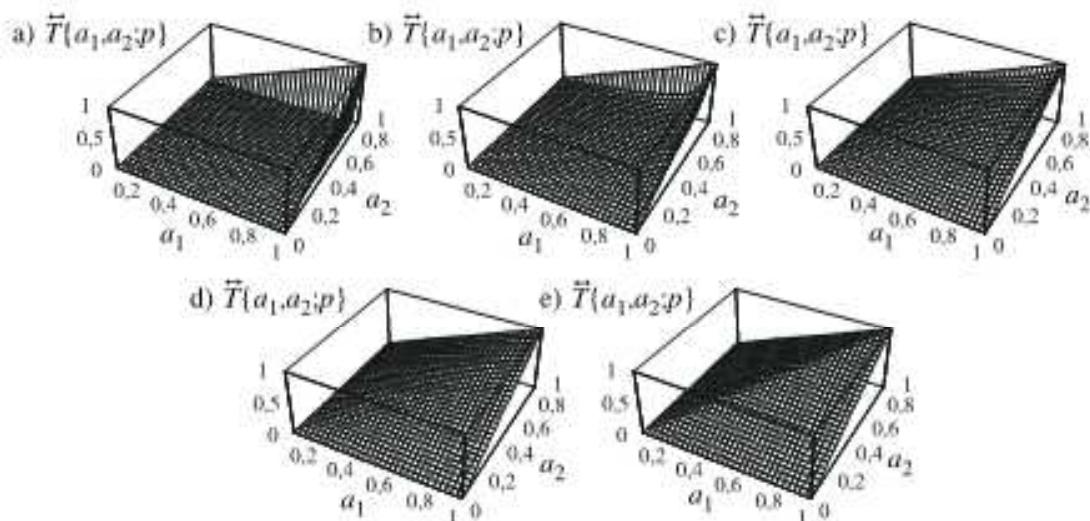
Do konstrukcji systemów elastycznych można także wykorzystać parametryzowane odmiany norm trójkątnych. Należą do nich m.in. normy trójkątne Dombi, Hamachera, Yagera, Franka, Webera, Dubois i Pradego, Schweizera oraz Mizumoto. Do ich zapisu będą stosowane notacje $\tilde{T}\{a_1, a_2, \dots, a_n; p\}$ oraz $\tilde{S}\{a_1, a_2, \dots, a_n; p\}$. Parametryzowane normy trójkątne charakteryzują się tym, że odpowiadające im hiperpłaszczyzny mogą być modyfikowane w wyniku uczenia parametru p .

Przykład 10.3

Parametryzowana t -norma Dombi jest określona następująco:

$$\tilde{T}\{\mathbf{a}; p\} = \begin{cases} t\text{-norma graniczna} & \text{dla } p = 0, \\ \left(1 + \left(\sum_{i=1}^n \left(\frac{1-a_i}{a_i}\right)^p\right)^{\frac{1}{p}}\right)^{-1} & \text{dla } p \in (0, \infty), \\ t\text{-norma Zadeha} & \text{dla } p = \infty. \end{cases} \quad (10.8)$$

Jej działanie dla $n = 2$ ilustruje rysunek 10.4.



Rys. 10.4. Hiperpłaszczyzny funkcji (10.8) dla a) $p = 0,10$, b) $p = 0,25$, c) $p = 0,50$, d) $p = 1,00$, e) $p = 10,00$

Parametryzowana t -konorma Dombi jest określona następująco:

$$\tilde{S}\{\mathbf{a}; p\} = \begin{cases} t\text{-konorma graniczna} & \text{dla } p = 0, \\ 1 - \left(1 + \left(\sum_{i=1}^n \left(\frac{a_i}{1-a_i}\right)^p\right)^{\frac{1}{p}}\right)^{-1} & \text{dla } p \in (0, \infty), \\ t\text{-konorma Zadeha} & \text{dla } p = \infty. \end{cases} \quad (10.9)$$

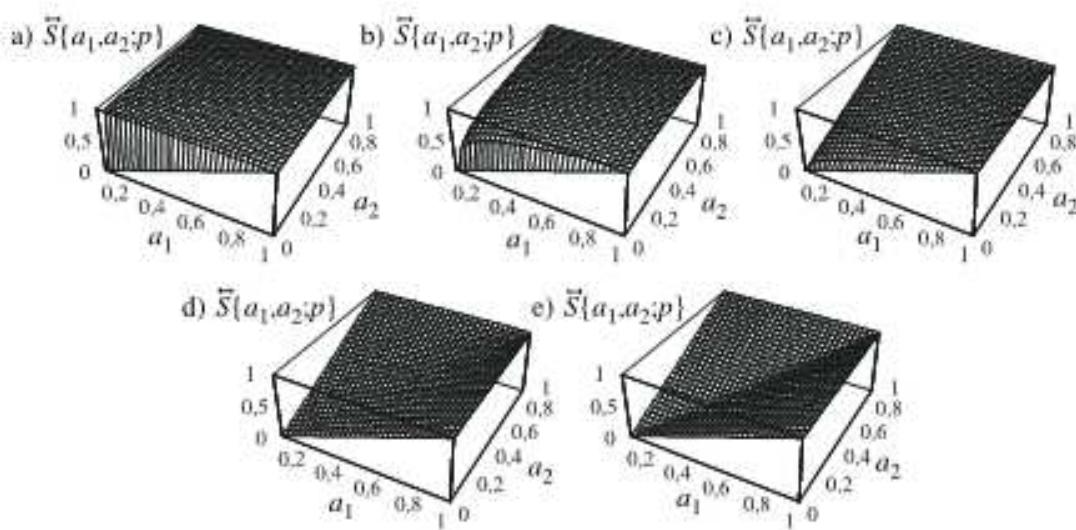
Rysunek 10.5 ilustruje jej działanie.

Addytywny generator parametryzowanej t -normy Dombi przybiera postać

$$t_{\text{add}}(x) = \left(\frac{1-x}{x}\right)^p, \quad (10.10)$$

a addytywny generator parametryzowanej t -konormy Dombi jest określony następująco:

$$s_{\text{add}}(x) = \left(\frac{x}{1-x}\right)^p. \quad (10.11)$$

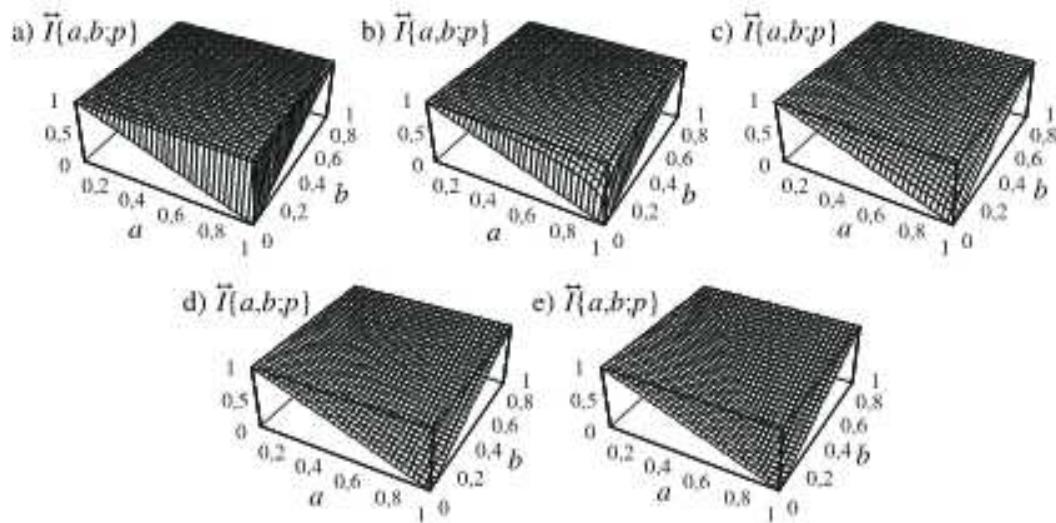


Rys. 10.5. Hiperplaszczyzny funkcji (10.9) dla a) $p = 0,10$, b) $p = 0,25$, c) $p = 0,50$, d) $p = 1,00$, e) $p = 10,00$

Parametryzowana t -norma Dombi dla $n = 2$ może odgrywać rolę „implikacji inżynierskiej”. Łącząc ideę parametryzowanej t -konormy Dombi z ideą S -implikacji, otrzymuje się parametryzowaną S -implikację Dombi, którą zapisuje się następująco:

$$\tilde{I}(a, b; p) = 1 - \left(1 + \left(\left(\frac{1-a}{a} \right)^p + \left(\frac{b}{1-b} \right)^p \right)^{\frac{1}{p}} \right)^{-1} \quad (10.12)$$

dla $p \in (0, \infty)$. Działanie parametryzowanej S -implikacji Dombi ilustruje rysunek 10.6.



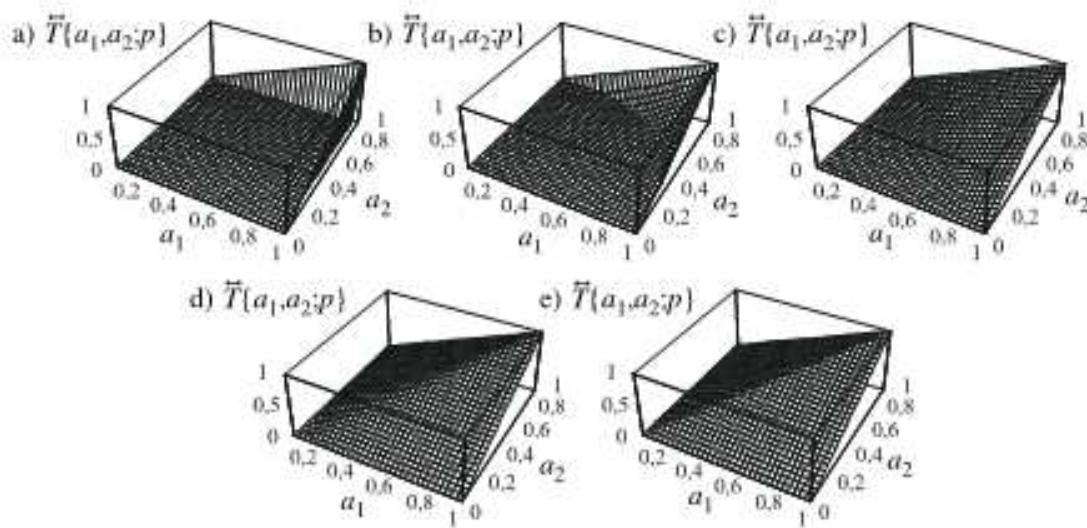
Rys. 10.6. Hiperplaszczyzny funkcji (10.12) dla a) $p = 0,10$, b) $p = 0,25$, c) $p = 0,50$, d) $p = 1,00$, e) $p = 10,00$

Przykład 10.4

Parametryzowana t -norma Yagera jest określona następująco:

$$\tilde{T}\{\mathbf{a}; p\} = \begin{cases} t\text{-norma graniczna} & \text{dla } p = 0, \\ \max \left\{ 0, 1 - \left(\sum_{i=1}^n (1 - a_i)^p \right)^{\frac{1}{p}} \right\} & \text{dla } p \in (0, \infty), \\ t\text{-norma Zadeha} & \text{dla } p = \infty \end{cases} \quad (10.13)$$

dla $p > 0$. Jej działanie dla $n = 2$ ilustruje rysunek 10.7.

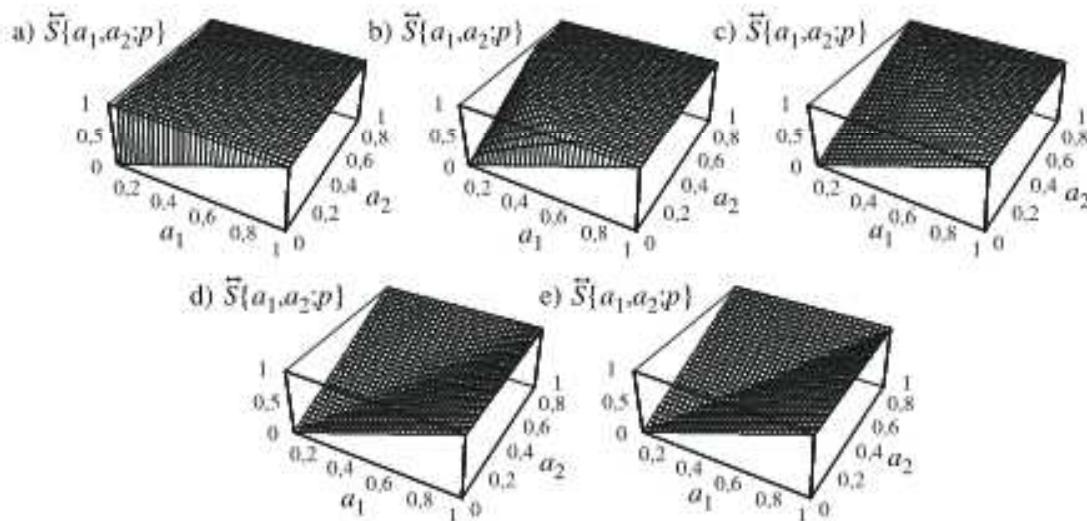


Rys. 10.7. Hiperplane funkcji (10.13) dla a) $p = 0,1$, b) $p = 0,5$, c) $p = 1,0$, d) $p = 10,0$, e) $p = 100,0$

Parametryzowana t -konorma Yagera jest określona następująco:

$$\tilde{S}[\mathbf{a}; p] = \begin{cases} t\text{-konorma graniczna} & \text{dla } p = 0, \\ \min \left\{ 1, \left(\sum_{i=1}^n (a_i)^p \right)^{\frac{1}{p}} \right\} & \text{dla } p \in (0, \infty), \\ t\text{-konorma Zadeha} & \text{dla } p = \infty. \end{cases} \quad (10.14)$$

Rysunek 10.8 ilustruje jej działanie.



Rys. 10.8. Hiperplane funkcji (10.14) dla a) $p = 0,1$, b) $p = 0,5$, c) $p = 1,0$, d) $p = 10,0$, e) $p = 100,0$

Addytywny generator parametryzowanej t -normy Yagera przybiera postać

$$t_{\text{add}}(x) = (1 - x)^p, \quad (10.15)$$

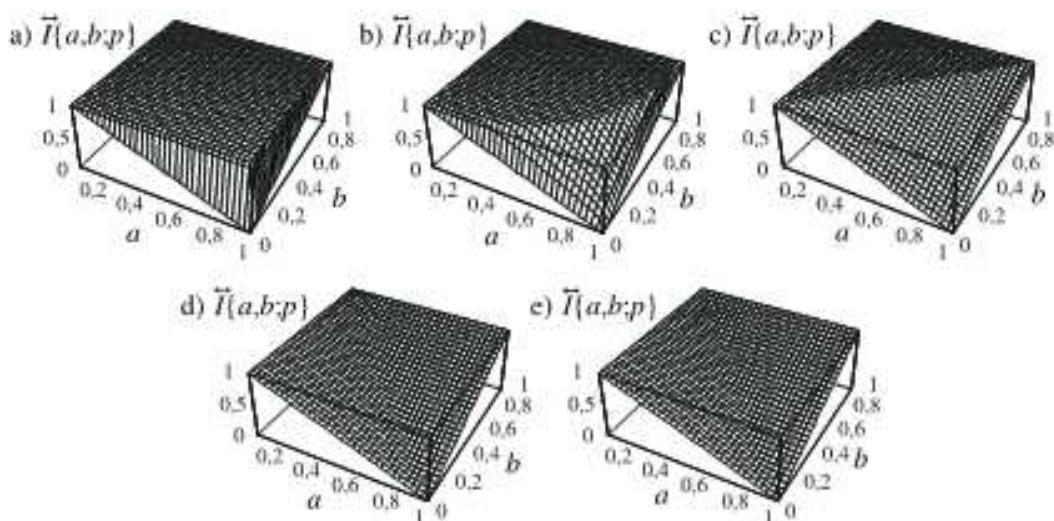
a addytywny generator parametryzowanej t -konormy Yagera jest określony następująco:

$$s_{\text{add}}(x) = x^p. \quad (10.16)$$

Parametryzowana t -norma Yagera dla $n = 2$ może pełnić funkcję „implikacji inżynierskiej”. Łącząc ideę parametryzowanej t -konormy Yagera z ideą S -implikacji, otrzymuje się parametryzowaną S -implikację Yagera, którą zapisuje się następująco:

$$\tilde{I}(a, b; p) = \min \left\{ 1, \left((1 - a)^p + b^p \right)^{\frac{1}{p}} \right\}. \quad (10.17)$$

Działanie parametryzowanej S -implikacji Yagera ilustruje rysunek 10.9.



Rys. 10.9. Hiperplaszczyzny funkcji (10.17) dla a) $p = 0,1$, b) $p = 0,5$, c) $p = 1,0$, d) $p = 10,0$, e) $p = 100,0$

Do konstrukcji systemów typu Mamdaniego można użyć następujących parametryzowanych norm trójkątnych:

- $\tilde{T}_1\{a_1, a_2, \dots, a_n; p^r\}$ do agregacji przesłanek w poszczególnych regułach;
- $\tilde{T}_2\{b_1, b_2; p^l\}$ do połączenia przesłanek i następników reguł;
- $\tilde{S}\{c_1, c_2, \dots, c_N; p^{agr}\}$ do agregacji reguł,

gdzie n jest liczbą wejść oraz N jest liczbą reguł.

W konstrukcji systemów typu logicznego wykorzystujących S -implikację można użyć następujących parametryzowanych norm trójkątnych:

- $\tilde{T}_1\{a_1, a_2, \dots, a_n; p^r\}$ do agregacji przesłanek w poszczególnych regułach;
- $\tilde{S}\{1 - b_1, b_2; p^l\}$ do połączenia przesłanek i następników reguł;
- $\tilde{T}_2\{c_1, c_2, \dots, c_N; p^{agr}\}$ do agregacji reguł,

gdzie n jest liczbą wejść oraz N jest liczbą reguł.

Należy podkreślić, że parametry p^r , p^l i p^{agr} można znaleźć na drodze uczenia.

10.4. Przełączane normy trójkątne

Skonstruujemy funkcję $H(\mathbf{a}; v)$, która w zależności od wartości parametru v przybiera postać t -normy lub t -konormy. Do konstrukcji tej funkcji wykorzystamy poniżej zdefiniowany operator kompromisowości.

DEFINICJA 10.1

Funkcję

$$\tilde{N}_v : [0, 1] \rightarrow [0, 1] \quad (10.18)$$

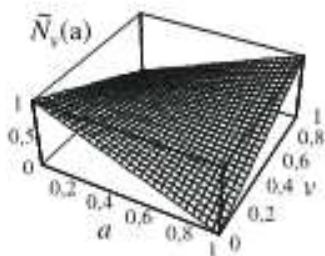
określoną jako

$$\tilde{N}_v(a) = (1 - v)N(a) + va \quad (10.19)$$

nazywamy *operatorem kompromisowości*, przy czym $v \in [0, 1]$ i $N(a) = \tilde{N}_0(a) = 1 - a$. Można zauważyc, że $\tilde{N}_{1-v}(a) = \tilde{N}_v(1 - a) = 1 - \tilde{N}_v(a)$ oraz

$$\tilde{N}_v(a) = \begin{cases} N(a) & \text{dla } v = 0, \\ \frac{1}{2} & \text{dla } v = \frac{1}{2}, \\ a & \text{dla } v = 1. \end{cases} \quad (10.20)$$

Funkcja \tilde{N}_v dla $v = 0$ jest *negacją typu strong*. Jej działanie ilustruje rysunek 10.10.



Rys. 10.10. Ilustracja działania operatora kompromisowości (10.19)

DEFINICJA 10.2

Funkcje

$$H : [0, 1]^n \rightarrow [0, 1] \quad (10.21)$$

określona jako

$$H(\mathbf{a}; v) = \tilde{N}_v \left(\sum_{i=1}^n \{\tilde{N}_v(a_i)\} \right) = \tilde{N}_{1-v} \left(\sum_{i=1}^n \{\tilde{N}_{1-v}(a_i)\} \right) \quad (10.22)$$

nazywamy *H-funkcją*, przy czym $v \in [0, 1]$.

TWIERDZENIE 10.1

Niech T i S będą normami trójkątnymi dualnymi. Wówczas funkcja H , zdefiniowana wzorem (10.22), zmienia swój kształt od t -normy do t -konormy, gdy v zmienia się od 0 do 1.

Dowód. Z założenia wynika, że

$$T\{\mathbf{a}\} = N(S\{N(a_1), N(a_2), \dots, N(a_n)\}). \quad (10.23)$$

Dla $v = 0$ wzór (10.23) może być zapisany następująco:

$$T\{\mathbf{a}\} = \tilde{N}_0(S\{\tilde{N}_0(a_1), \tilde{N}_0(a_2), \dots, \tilde{N}_0(a_n)\}). \quad (10.24)$$

Jednocześnie

$$S\{\mathbf{a}\} = \tilde{N}_1(S\{\tilde{N}_1(a_1), \tilde{N}_1(a_2), \dots, \tilde{N}_1(a_n)\}) \quad (10.25)$$

dla $v = 1$. Prawe strony wzorów (10.24) i (10.25) mogą być zapisane następująco:

$$H(\mathbf{a}; v) = \tilde{N}_v \left(\sum_{i=1}^n \{\tilde{N}_v(a_i)\} \right) \quad (10.26)$$

dla, odpowiednio, $v = 0$ i $v = 1$. Jeśli parametr v zmienia swoją wartość od 0 do 1, to funkcja H jest płynnie przełączana między t -normą i t -konormą. Łatwo zauważyć, że

$$H(\mathbf{a}; v) = \begin{cases} T\{\mathbf{a}\} & \text{dla } v = 0, \\ \frac{1}{2} & \text{dla } v = \frac{1}{2}, \\ S\{\mathbf{a}\} & \text{dla } v = 1. \end{cases} \quad (10.27)$$

Przykład 10.5

Przełączana H -funkcja skonstruowana z wykorzystaniem t -normy lub t -konormy Zadeha przybiera postać

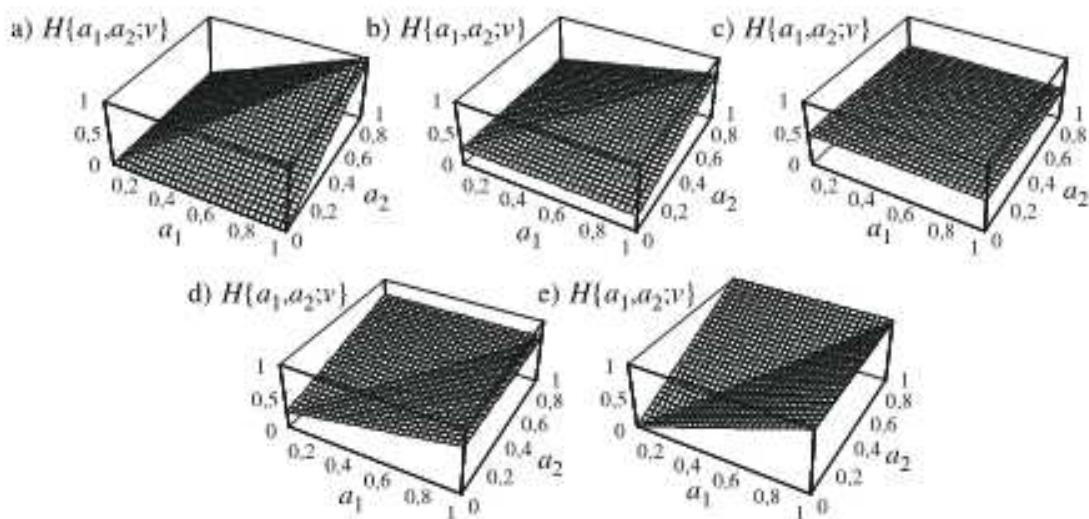
$$\begin{aligned} H(a_1, a_2; v) &= \tilde{N}_{1-v}(\min\{\tilde{N}_{1-v}(a_1), \tilde{N}_{1-v}(a_2)\}) \\ &= \tilde{N}_v(\max\{\tilde{N}_v(a_1), \tilde{N}_v(a_2)\}), \end{aligned} \quad (10.28)$$

przy czym v zmienia się od wartości 0 do 1. Łatwo zauważyć, że

$$H(a_1, a_2; 0) = T\{a_1, a_2\} = \min\{a_1, a_2\}, \quad (10.29)$$

$$H(a_1, a_2; 1) = S\{a_1, a_2\} = \max\{a_1, a_2\}. \quad (10.30)$$

Działanie H -funkcji Zadeha ilustruje rysunek 10.11.

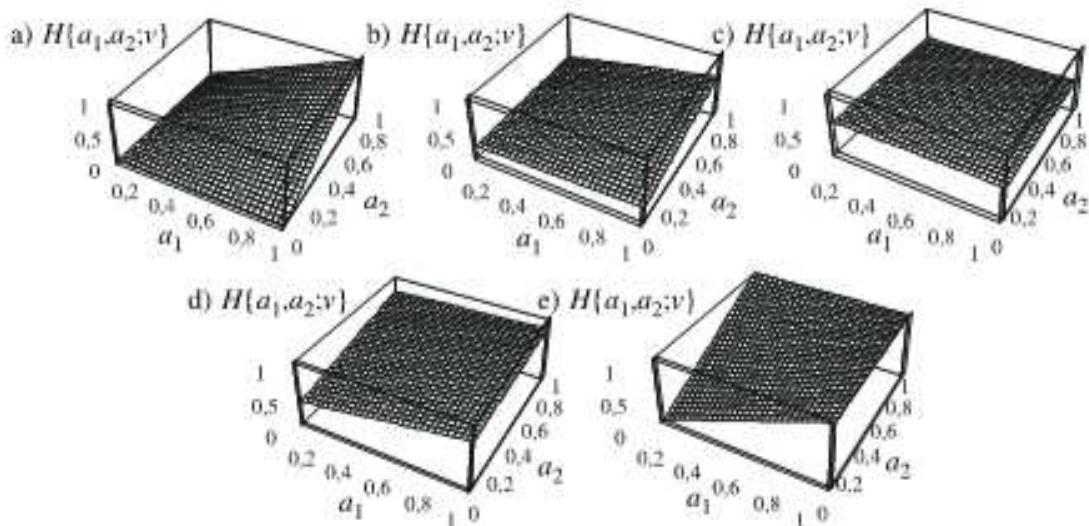


Rys. 10.11. Hiperplaszczyzny funkcji (10.28) dla a) $v = 0,00$, b) $v = 0,15$, c) $v = 0,50$, d) $v = 0,85$, e) $v = 1,00$

Przykład 10.6

Przełączana H -funkcja skonstruowana z wykorzystaniem t -normy lub t -konormy algebraicznej przybiera postać

$$\begin{aligned} H(a_1, a_2; v) &= \tilde{N}_{1-v}(\tilde{N}_{1-v}(a_1)\tilde{N}_{1-v}(a_2)) \\ &= \tilde{N}_v(1 - (1 - \tilde{N}_v(a_1))(1 - \tilde{N}_v(a_2))), \end{aligned} \quad (10.31)$$



Rys. 10.12. Hiperplaszczyzny funkcji (10.31) dla a) $v = 0,00$, b) $v = 0,15$, c) $v = 0,50$, d) $v = 0,85$, e) $v = 1,00$

przy czym v zmienia się od wartości 0 do 1. Łatwo zauważyc, że

$$T\{a_1, a_2\} = H(a_1, a_2; 0) = a_1 a_2, \quad (10.32)$$

$$S\{a_1, a_2\} = H(a_1, a_2; 1) = a_1 + a_2 - a_1 a_2. \quad (10.33)$$

Działanie H -funkcji algebraicznej ilustruje rysunek 10.12.

Skonstruujemy teraz tzw. H -implikację, która może się przełączac między „implikacją inżynierską” (t -norma) i implikacją rozmytą (S -implikacją).

TWIERDZENIE 10.2

Niech T i S będą normami trójkątnymi dualnymi. Wówczas H -implikacja zdefiniowana następująco:

$$I(a, b; v) = H(\tilde{N}_{1-v}(a), b; v) \quad (10.34)$$

zmienia się od „implikacji inżynierskiej”

$$I_{\text{eng}}(a, b) = I(a, b; 0) = T\{a, b\} \quad (10.35)$$

do implikacji rozmytej

$$I_{\text{fuzzy}}(a, b) = I(a, b; 1) = S\{1 - a, b\}, \quad (10.36)$$

gdy parametr v zmienia swoją wartość od 0 do 1.

Dowód. Twierdzenie 10.2 jest bezpośrednim następstwem twierdzenia 10.1.

Przykład 10.7

Przełączaną H -implikację, która może się zmieniać między „implikacją inżynierską” wyrażoną przez t -normę Zadeha

$$\begin{aligned} I_{\text{eng}}(a, b) &= H(a, b; 0) \\ &= T\{a, b\} \\ &= \min\{a, b\} \end{aligned} \quad (10.37)$$

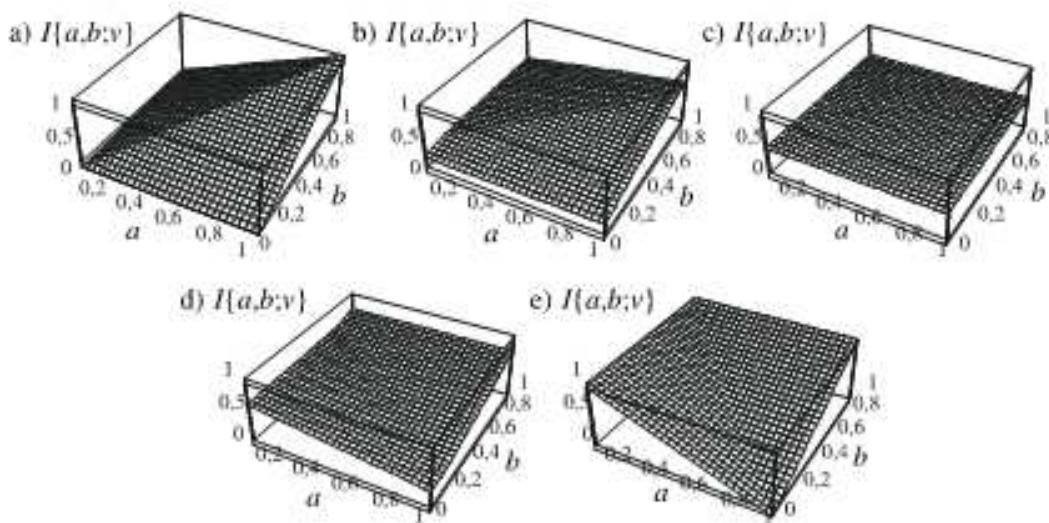
a S -implikacją binarną

$$\begin{aligned} I_{\text{fuzzy}}(a, b) &= H(\tilde{N}_0(a), b; 1) \\ &= S\{N(a), b\} \\ &= \max\{N(a), b\}, \end{aligned} \quad (10.38)$$

można wyrazić następująco:

$$I(a, b; v) = H(\tilde{N}_{1-v}(a), b; v), \quad (10.39)$$

przy czym v zmienia się od 0 do 1. Działanie H -implikacji danej wzorem (10.39) ilustruje rysunek 10.13.



Rys. 10.13. Hiperpłaszczyzny funkcji (10.39) dla a) $v = 0,00$, b) $v = 0,15$, c) $v = 0,50$, d) $v = 0,85$, e) $v = 1,00$

Przykład 10.8

Przelączaną H -implikacją, która może się zmieniać między „implikacją inżynierską” wyrażoną przez t -normę algebraiczną

$$\begin{aligned} I_{\text{eng}}(a, b) &= H(a, b; 0) \\ &= T\{a, b\} \\ &= ab \end{aligned} \quad (10.40)$$

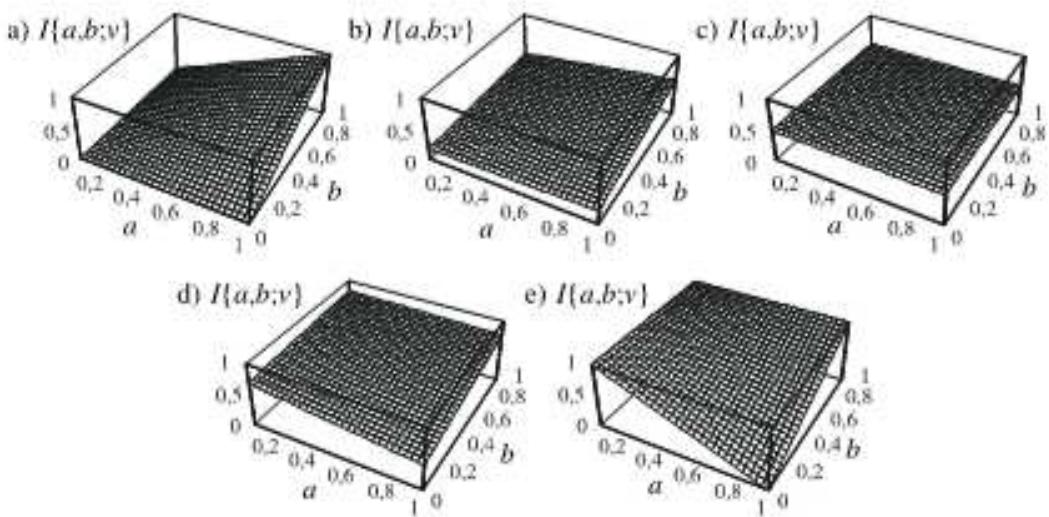
a S -implikacją binarną

$$\begin{aligned} I_{\text{fuzzy}}(a, b) &= H(\tilde{N}_0(a), b; 1) \\ &= S\{N(a), b\} \\ &= 1 - a + ab, \end{aligned} \quad (10.41)$$

można wyrazić następująco:

$$I(a, b; v) = H(\tilde{N}_{1-v}(a), b; v), \quad (10.42)$$

przy czym v zmienia się od 0 do 1. Działanie H -implikacji danej wzorem (10.42) ilustruje rysunek 10.14.



Rys. 10.14. Hiperpłaszczyzny funkcji (10.42) dla a) $v = 0,00$, b) $v = 0,15$, c) $v = 0,50$, d) $v = 0,85$, e) $v = 1,00$

10.5. Systemy elastyczne

Korzystając z koncepcji przełączanych norm trójkątnych oraz przełączanych implikacji, skonstruujemy system neuronowo-rozmyty, którego struktura może się zmieniać między systemem typu Mamdaniego a systemem typu logicznego.

TWIERDZENIE 10.3

Niech T i S będą normami trójkątnymi dualnymi. Wówczas system neuronowo-rozmyty

$$\tau_k(\bar{\mathbf{x}}) = H \left(\begin{array}{c} \mu_{A_1^k}(\bar{x}_1), \dots, \mu_{A_n^k}(\bar{x}_n); \\ 0 \end{array} \right), \quad (10.43)$$

$$I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r) = H \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r); \\ v \end{array} \right), \quad (10.44)$$

$$\text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r) = H \left(\begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ 1-v \end{array} \right), \quad (10.45)$$

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot \text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r)}{\sum_{r=1}^N \text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r)} \quad (10.46)$$

zmienia się między systemem typu Mamdaniego ($v = 0$) i systemem typu logicznego ($v = 1$) wraz ze zmianą parametru v od 0 do 1.

Dowód. Dla $v = 0$ wzór (10.46) przybiera postać

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot S_{k=1}^N \{ T \{ \tau_k(\bar{\mathbf{x}}), \mu_{B^k}(\bar{y}^r) \} \}}{\sum_{r=1}^N S_{k=1}^N \{ T \{ \tau_k(\bar{\mathbf{x}}), \mu_{B^k}(\bar{y}^r) \} \}}. \quad (10.47)$$

Latwo zauważyć, że powyższy wzór opisuje system typu Mamdaniego. Dla $v = 1$ mamy

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot T_{k=1}^N \{ S \{ N(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r) \} \}}{\sum_{r=1}^N T_{k=1}^N \{ S \{ N(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r) \} \}}. \quad (10.48)$$

Zależność (10.48) opisuje system typu logicznego z wykorzystaniem S -implikacji. Dla wartości parametru $v \in (0, 1)$ wnioskowanie przebiega zgodnie z definicją H -implikacji, co kończy dowód.

W tabeli 10.1 przedstawiono operatory implikacji i agregacji dla zmieniającego się parametru v .

Tabela 10.1. Operatory implikacji i agregacji dla zmieniającego się parametru v

Parametr v	Implikacja	Agregacja
$v = 0$	$T\{a, b\}$	t -konorma
$v = 1$	$S\{1 - a, b\}$	t -norma
$0 < v < 1$	$H(\tilde{N}_{1-v}(a), b; v)$	$H(a, b; 1 - v)$
$v = 0,5$	$H(a, b; 0,5) = 0,5$	$H(a, b; 0,5) = 0,5$

System opisany zależnościami (10.43)–(10.46) jest systemem elastycznym, gdyż umożliwia dobór modelu wnioskowania w wyniku procesu uczenia. Jednak system ten nie uwzględnia pozostałych aspektów elastyczności opisanych w podrozdziałach 10.2 i 10.3. Obecnie do systemu (10.46) danego w twierdzeniu 10.3 wprowadzimy koncepcję miękkich norm trójkątnych, parametryzowanych norm trójkątnych oraz wag reguł i wag przesłanek reguł. Wówczas elastyczny system neuronowo-rozmyty przybiera następującą postać:

$$\tau_k(\bar{\mathbf{x}}) = \begin{pmatrix} (1 - \alpha^r) \text{avg}(\mu_{A_1^r}(\bar{x}_1), \dots, \mu_{A_n^r}(\bar{x}_n)) + \\ + \alpha^r \tilde{H}^* \left(\begin{array}{c} \mu_{A_1^r}(\bar{x}_1), \dots, \mu_{A_n^r}(\bar{x}_n); \\ w_{1,k}^r, \dots, w_{n,k}^r, p^r, 0 \end{array} \right) \end{pmatrix}, \quad (10.49)$$

$$I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r) = \begin{pmatrix} (1 - \alpha^l) \text{avg}(\tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^r}(\bar{y}^r)) + \\ + \alpha^l \tilde{H} \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^r}(\bar{y}^r); \\ p^l, v \end{array} \right) \end{pmatrix}, \quad (10.50)$$

$$\text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r) = \begin{pmatrix} (1 - \alpha^{\text{agr}}) \text{avg}(I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r)) + \\ + \alpha^{\text{agr}} \tilde{H}^* \left(\begin{array}{c} I_{1,r}(\bar{\mathbf{x}}, \bar{y}^r), \dots, I_{N,r}(\bar{\mathbf{x}}, \bar{y}^r); \\ w_1^{\text{agr}}, \dots, w_N^{\text{agr}}, p^{\text{agr}}, 1 - v \end{array} \right) \end{pmatrix}. \quad (10.51)$$

W systemie opisanym zależnościami (10.46) i (10.49)–(10.51) można wyróżnić następujące parametry:

- $v \in [0, 1]$, parametr typu modelu wnioskowania,
- $\alpha^r \in [0, 1]$, $\alpha^l \in [0, 1]$, $\alpha^{\text{agr}} \in [0, 1]$, parametry elastyczności w sensie Yagera i Fileva operatorów agregacji przesłanek, operatorów wnioskowania i operatorów agregacji reguł,
- $p^r \in [0, \infty)$, $p^l \in [0, \infty)$, $p^{\text{agr}} \in [0, \infty)$, parametry kształtu hiperpłaszczyzn operatorów agregacji przesłanek, operatorów wnioskowania i operatorów agregacji reguł,
- $w_{i,k}^r \in [0, 1]$, $i = 1, \dots, n$, $k = 1, \dots, N$, wagi przesłanek reguł,
- $w_k^{\text{agr}} \in [0, 1]$, $k = 1, \dots, N$, wagi reguł,
- $p_{u,i,k}^A$, $u = 1, 2, \dots, P^A$, $i = 1, 2, \dots, n$, parametry kształtu funkcji przynależności wejściowych zbiorów rozmytych,
- $p_{1,k}^B = \bar{y}^k$, $k = 1, 2, \dots, N$, centra funkcji przynależności wyjściowych zbiorów rozmytych,
- $p_{u,k}^B$, $u = 2, 3, \dots, P^B$, $k = 1, 2, \dots, N$, parametry kształtu funkcji przynależności wyjściowych zbiorów rozmytych.

Wyżej wymienione parametry będą podlegały uczeniu w następnym podrozdziale.

10.6. Algorytmy uczenia

Wyprowadzimy teraz gradientowe algorytmy uczenia systemu opisanego zależnościami (10.46) i (10.49)–(10.51). Parametry te modyfikuje się w sposób iteracyjny zgodnie z poniższymi zależnościami:

$$v(t+1) = v(t) - \eta \Delta v(t), \quad (10.52)$$

$$\alpha^r(t+1) = \alpha^r(t) - \eta \Delta \alpha^r(t), \quad (10.53)$$

$$\alpha^I(t+1) = \alpha^I(t) - \eta \Delta \alpha^I(t), \quad (10.54)$$

$$\alpha^{agr}(t+1) = \alpha^{agr}(t) - \eta \Delta \alpha^{agr}(t), \quad (10.55)$$

$$p^r(t+1) = p^r(t) - \eta \Delta p^r(t), \quad (10.56)$$

$$p^I(t+1) = p^I(t) - \eta \Delta p^I(t), \quad (10.57)$$

$$p^{agr}(t+1) = p^{agr}(t) - \eta \Delta p^{agr}(t), \quad (10.58)$$

$$w_{i,k}^r(t+1) = w_{i,k}^r(t) - \eta \Delta w_{i,k}^r(t), \quad (10.59)$$

$$w_k^{agr}(t+1) = w_k^{agr}(t) - \eta \Delta w_k^{agr}(t), \quad (10.60)$$

$$p_{u,i,k}^A(t+1) = p_{u,i,k}^A(t) - \eta \Delta p_{u,i,k}^A(t), \quad (10.61)$$

$$p_{u,k}^B(t+1) = p_{u,k}^B(t) - \eta \Delta p_{u,k}^B(t); \quad u = 2, \dots, P^B, \quad (10.62)$$

$$\bar{y}^r(t+1) = p_{1,r}^B(t+1) = \bar{y}^r(t) - \eta \Delta \bar{y}^r(t). \quad (10.63)$$

Członny Δ w powyższych zależnościach określa się następująco:

$$\Delta v = \sum_{k=1}^N \sum_{r=1}^N \varepsilon_{k,r}^I \{v\} + \sum_{r=1}^N \varepsilon_r^{agr} \{v\}, \quad (10.64)$$

$$\Delta \alpha^r = \sum_{k=1}^N \varepsilon_k^r \{\alpha^r\}, \quad (10.65)$$

$$\Delta \alpha^I = \sum_{k=1}^N \sum_{r=1}^N \varepsilon_{k,r}^I \{\alpha^I\}, \quad (10.66)$$

$$\Delta \alpha^{agr} = \sum_{r=1}^N \varepsilon_r^{agr} \{\alpha^{agr}\}, \quad (10.67)$$

$$\Delta p^r = \sum_{k=1}^N \varepsilon_k^r \{p^r\}, \quad (10.68)$$

$$\Delta p^I = \sum_{k=1}^N \sum_{r=1}^N \varepsilon_{k,r}^I \{p^I\}, \quad (10.69)$$

$$\Delta p^{agr} = \sum_{r=1}^N \varepsilon_r^{agr} \{p^{agr}\}, \quad (10.70)$$

$$\Delta w_{i,k}^r = \varepsilon_k^r \{w_{i,k}^r\}, \quad (10.71)$$

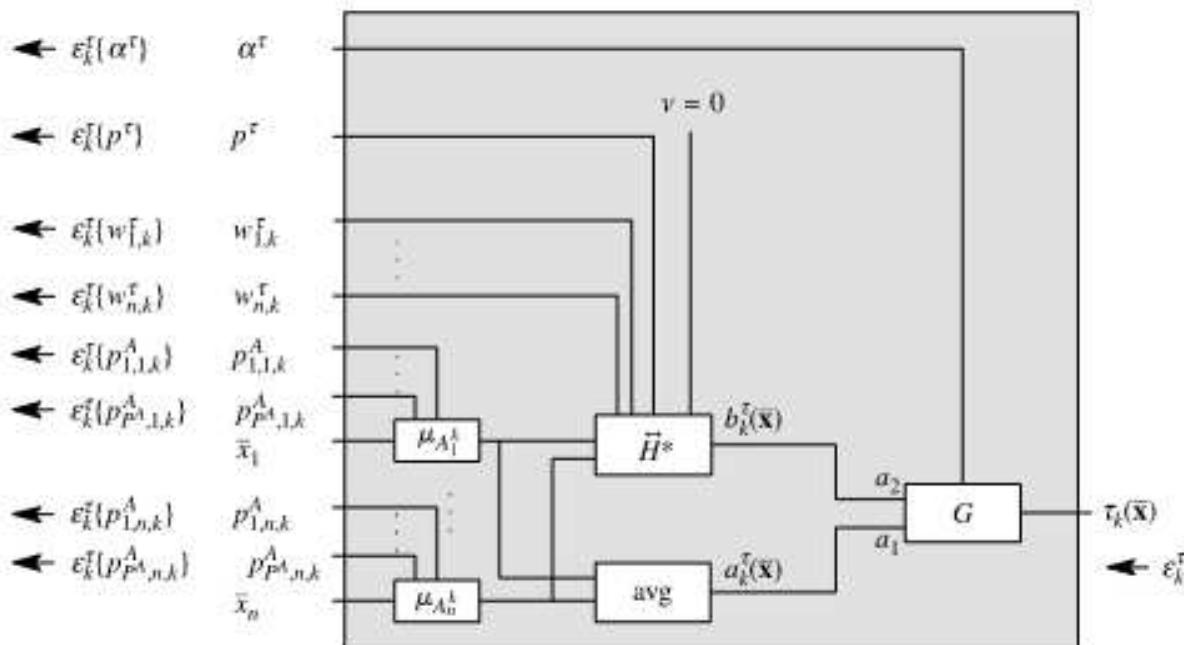
$$\Delta w_k^{agr} = \sum_{r=1}^N \varepsilon_r^{agr} \{w_k^{agr}\}, \quad (10.72)$$

$$\Delta p_{u,i,k}^A = \varepsilon_k^r \{p_{u,i,k}^A\}, \quad (10.73)$$

$$\Delta p_{u,k}^B = \sum_{r=1}^N \varepsilon_{k,r}^I \{p_{u,k}^B\}; \quad u = 2, \dots, P^B, \quad (10.74)$$

$$\Delta \bar{y}^r = \Delta p_{1,r}^B = \varepsilon^{\text{def}} \{\bar{y}^r\} + \sum_{k=1}^N \varepsilon_{k,r}^I \{\bar{y}^r\} + \sum_{k=1}^N \varepsilon_{r,k}^I \{p_{1,r}^B\}. \quad (10.75)$$

Błędy propagowane przez poszczególne warstwy systemu określają się analogicznie jak w przypadku algorytmów uczenia dotyczących systemów nieelastycznych, które opisano w podrozdziale 9.6. Sposób propagacji błędów pokazano na rysunku 9.11.



Rys. 10.15. Blok poziomu aktywności reguły systemu elastycznego

Błędy propagowane przez bloki poziomów aktywności reguł określają się następująco (rys. 10.15):

$$\varepsilon_k^r \{ \alpha^r \} = \varepsilon_k^r \frac{\partial \tau_k(\bar{x})}{\partial \alpha^r}, \quad (10.76)$$

$$\varepsilon_k^r \{ p^r \} = \varepsilon_k^r \frac{\partial \tau_k(\bar{x})}{\partial b_k^r(\bar{x})} \frac{\partial b_k^r(\bar{x})}{\partial p^r}, \quad (10.77)$$

$$\varepsilon_k^r \{ w_{i,k}^r \} = \varepsilon_k^r \frac{\partial \tau_k(\bar{x})}{\partial b_k^r(\bar{x})} \frac{\partial b_k^r(\bar{x})}{\partial w_{i,k}^r}, \quad (10.78)$$

$$\varepsilon_k^r \{ p_{u,i,k}^A \} = \varepsilon_k^r \left(\frac{\frac{\partial \tau_k(\bar{x})}{\partial b_k^r(\bar{x})} \frac{\partial b_k^r(\bar{x})}{\partial \mu_{A_i^k}(\bar{x}_i)} +}{\frac{\partial \tau_k(\bar{x})}{\partial a_k^r(\bar{x})} \frac{\partial a_k^r(\bar{x})}{\partial \mu_{A_i^k}(\bar{x}_i)}} \right) \frac{\partial \mu_{A_i^k}(\bar{x}_i)}{\partial p_{u,i,k}^A}, \quad (10.79)$$

przy czym

$$\frac{\partial \tau_k(\bar{x})}{\partial a_k^r(\bar{x})} = \frac{\partial}{\partial a_k^r(\bar{x})} G \left(\frac{a_k^r(\bar{x}), b_k^r(\bar{x})}{\alpha^r} \right), \quad (10.80)$$

$$\frac{\partial \tau_k(\bar{x})}{\partial b_k^r(\bar{x})} = \frac{\partial}{\partial b_k^r(\bar{x})} G \left(\frac{a_k^r(\bar{x}), b_k^r(\bar{x})}{\alpha^r} \right), \quad (10.81)$$

$$\frac{\partial \tau_k(\bar{x})}{\partial \alpha^r} = \frac{\partial}{\partial \alpha^r} G \left(\frac{a_k^r(\bar{x}), b_k^r(\bar{x})}{\alpha^r} \right), \quad (10.82)$$

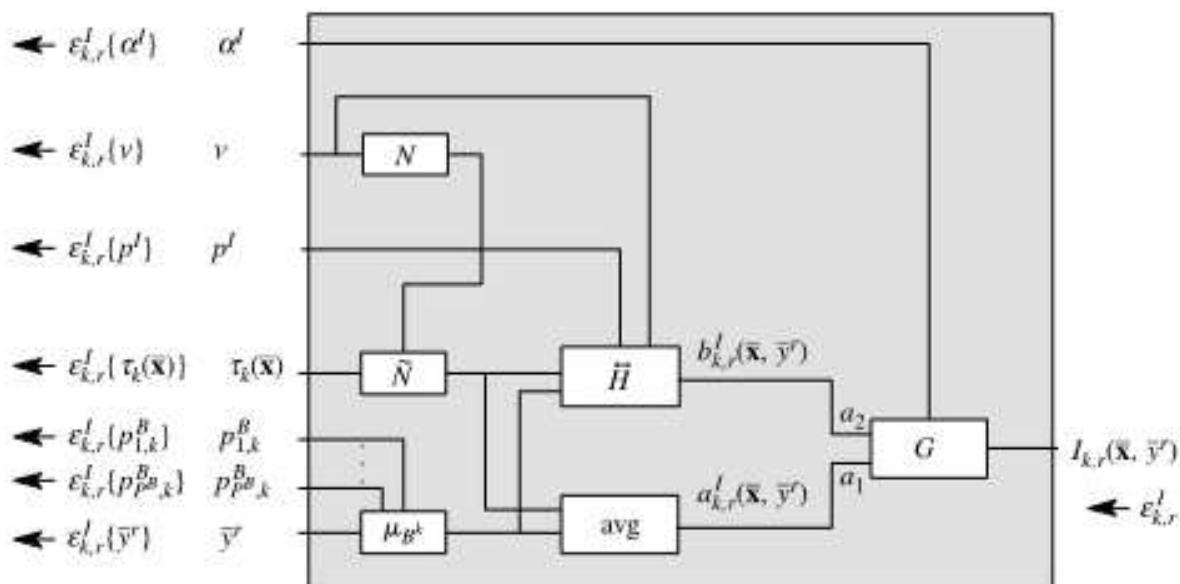
$$\frac{\partial a_k^r(\bar{x})}{\partial \mu_{A_i^k}(\bar{x}_i)} = \frac{\partial}{\partial \mu_{A_i^k}(\bar{x}_i)} \text{avg}(\mu_{A_1^k}(\bar{x}_1), \dots, \mu_{A_n^k}(\bar{x}_n)), \quad (10.83)$$

$$\frac{\partial b_k^r(\bar{\mathbf{x}})}{\partial p^r} = \frac{\partial}{\partial p^r} \tilde{H}^* \left(\begin{array}{c} \mu_{A_1^k}(\bar{x}_1), \dots, \mu_{A_n^k}(\bar{x}_n); \\ w_{1,k}^r, \dots, w_{n,k}^r, p^r, 0 \end{array} \right), \quad (10.84)$$

$$\frac{\partial b_k^r(\bar{\mathbf{x}})}{\partial w_{i,k}^r} = \frac{\partial}{\partial w_{i,k}^r} \tilde{H}^* \left(\begin{array}{c} \mu_{A_1^k}(\bar{x}_1), \dots, \mu_{A_n^k}(\bar{x}_n); \\ w_{1,k}^r, \dots, w_{n,k}^r, p^r, 0 \end{array} \right), \quad (10.85)$$

$$\frac{\partial b_k^r(\bar{\mathbf{x}})}{\partial \mu_{A_i^k}(\bar{x}_i)} = \frac{\partial}{\partial \mu_{A_i^k}(\bar{x}_i)} \tilde{H}^* \left(\begin{array}{c} \mu_{A_1^k}(\bar{x}_1), \dots, \mu_{A_n^k}(\bar{x}_n); \\ w_{1,k}^r, \dots, w_{n,k}^r, p^r, 0 \end{array} \right). \quad (10.86)$$

Występujące w powyższych zależnościach pochodne wyznacza się, korzystając ze wzorów zamieszczonych w dalszej części tego podrozdziału.



Rys. 10.16. Blok implikacji systemu elastycznego

Błędy propagowane przez bloki implikacji określają się następująco (rys. 10.16):

$$\varepsilon_{k,r}^I\{v\} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial v} + \\ + \left(\begin{array}{l} \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} + \\ + \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} \\ \times \frac{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))}{\partial(1-v)} \frac{\partial N(v)}{\partial v} \end{array} \right) \times \end{array} \right), \quad (10.87)$$

$$\varepsilon_{k,r}^I\{\alpha^I\} = \varepsilon_{k,r}^I \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \alpha^I}, \quad (10.88)$$

$$\varepsilon_{k,r}^I\{p^I\} = \varepsilon_{k,r}^I \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial p^I}, \quad (10.89)$$

$$\varepsilon_{k,r}^I\{p_{u,k}^B\} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} + \\ + \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} \end{array} \right) \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial p_{u,k}^B}, \quad (10.90)$$

$$\varepsilon_{k,r}^I\{\bar{y}^r\} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} + \\ + \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} \end{array} \right) \frac{\partial \mu_{B^k}(\bar{y}^r)}{\partial \bar{y}^r}, \quad (10.91)$$

$$\varepsilon_{k,r}^I\{\tau_k(\bar{\mathbf{x}})\} = \varepsilon_{k,r}^I \left(\begin{array}{l} \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} + \\ + \frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} \end{array} \right) \frac{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))}{\partial \tau_k(\bar{\mathbf{x}})}, \quad (10.92)$$

przy czym

$$\frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} = \frac{\partial}{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} G \left(\begin{array}{c} a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r), b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r); \\ \alpha^I \end{array} \right), \quad (10.93)$$

$$\frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} = \frac{\partial}{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)} G \left(\begin{array}{c} a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r), b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r); \\ \alpha^I \end{array} \right), \quad (10.94)$$

$$\frac{\partial I_{k,r}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \alpha^I} = \frac{\partial}{\partial \alpha^I} G \left(\begin{array}{c} a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r), b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r); \\ \alpha^I \end{array} \right), \quad (10.95)$$

$$\frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} = \frac{\partial}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} \text{avg}(\tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r)), \quad (10.96)$$

$$\frac{\partial a_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} = \frac{\partial}{\partial \mu_{B^k}(\bar{y}^r)} \text{avg}(\tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r)), \quad (10.97)$$

$$\frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial v} = \frac{\partial}{\partial v} \tilde{H} \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r); \\ p^I, v \end{array} \right), \quad (10.98)$$

$$\frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial p^I} = \frac{\partial}{\partial p^I} \tilde{H} \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r); \\ p^I, v \end{array} \right), \quad (10.99)$$

$$\frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} = \frac{\partial}{\partial \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}}))} \tilde{H} \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r); \\ p^I, v \end{array} \right), \quad (10.100)$$

$$\frac{\partial b_{k,r}^I(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \mu_{B^k}(\bar{y}^r)} = \frac{\partial}{\partial \mu_{B^k}(\bar{y}^r)} \tilde{H} \left(\begin{array}{c} \tilde{N}_{1-v}(\tau_k(\bar{\mathbf{x}})), \mu_{B^k}(\bar{y}^r); \\ p^I, v \end{array} \right). \quad (10.101)$$

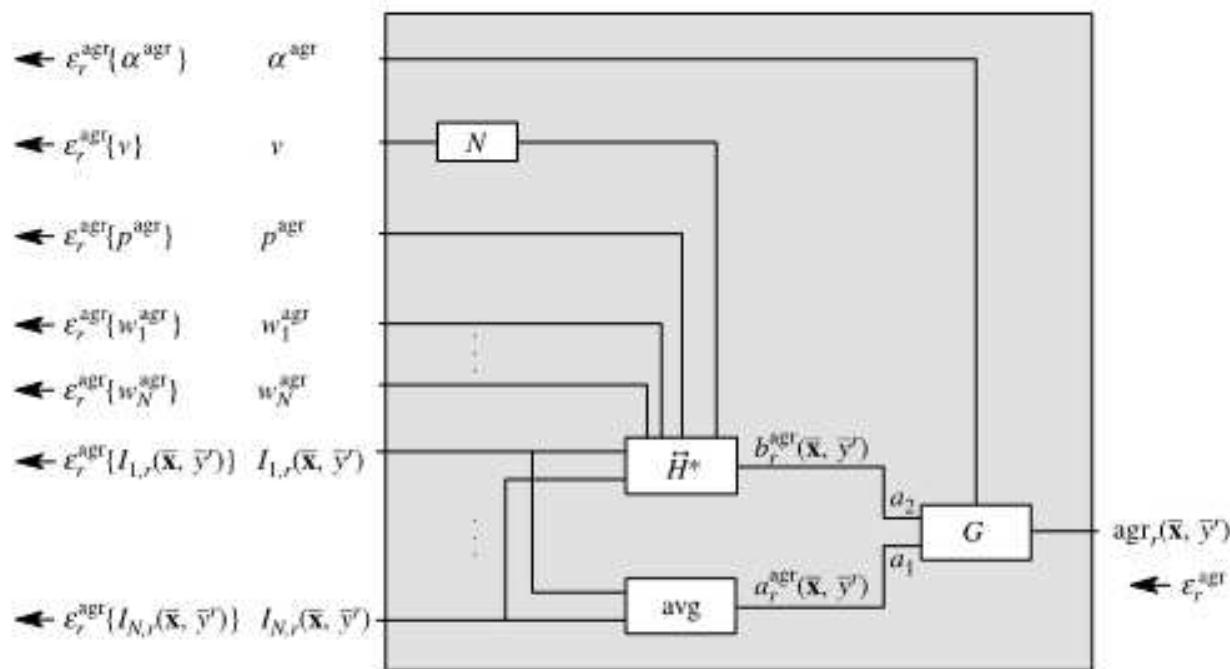
Występujące w powyższych zależnościach pochodne wyznacza się, korzystając ze wzorów zamieszczonych w dalszej części tego podrozdziału.

Błędy propagowane przez bloki agregacji określają się następująco (rys. 10.17):

$$\varepsilon_r^{\text{agr}}\{v\} = \varepsilon_r^{\text{agr}} \frac{\partial \text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_r^{\text{agr}}(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_r^{\text{agr}}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial (1-v)} \frac{\partial N(v)}{\partial v}, \quad (10.102)$$

$$\varepsilon_r^{\text{agr}}\{\alpha^{\text{agr}}\} = \varepsilon_r^{\text{agr}} \frac{\partial \text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r)}{\partial \alpha^{\text{agr}}}, \quad (10.103)$$

$$\varepsilon_r^{\text{agr}}\{p^{\text{agr}}\} = \varepsilon_r^{\text{agr}} \frac{\partial \text{agr}_r(\bar{\mathbf{x}}, \bar{y}^r)}{\partial b_r^{\text{agr}}(\bar{\mathbf{x}}, \bar{y}^r)} \frac{\partial b_r^{\text{agr}}(\bar{\mathbf{x}}, \bar{y}^r)}{\partial p^{\text{agr}}}, \quad (10.104)$$



Rys. 10.17. Blok agregacji systemu elastycznego

$$\varepsilon_r^{agr}\{w_k^{agr}\} = \varepsilon_r^{agr} \frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial b_r^{agr}(\bar{x}, \bar{y}^r)} \frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial w_k^{agr}}, \quad (10.105)$$

$$\varepsilon_r^{agr}\{I_{k,r}(\bar{x}, \bar{y}^r)\} = \varepsilon_r^{agr} \left(\frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial b_r^{agr}(\bar{x}, \bar{y}^r)} \frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} + \frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial a_r^{agr}(\bar{x}, \bar{y}^r)} \frac{\partial a_r^{agr}(\bar{x}, \bar{y}^r)}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} \right), \quad (10.106)$$

przy czym

$$\frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial a_r^{agr}(\bar{x}, \bar{y}^r)} = \frac{\partial}{\partial a_r^{agr}(\bar{x}, \bar{y}^r)} G \left(\frac{a_r^{agr}(\bar{x}, \bar{y}^r), b_r^{agr}(\bar{x}, \bar{y}^r)}{\alpha^{agr}} \right), \quad (10.107)$$

$$\frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial b_r^{agr}(\bar{x}, \bar{y}^r)} = \frac{\partial}{\partial b_r^{agr}(\bar{x}, \bar{y}^r)} G \left(\frac{a_r^{agr}(\bar{x}, \bar{y}^r), b_r^{agr}(\bar{x}, \bar{y}^r)}{\alpha^{agr}} \right), \quad (10.108)$$

$$\frac{\partial \text{agr}_r(\bar{x}, \bar{y}^r)}{\partial \alpha^{agr}} = \frac{\partial}{\partial \alpha^{agr}} G \left(\frac{a_r^{agr}(\bar{x}, \bar{y}^r), b_r^{agr}(\bar{x}, \bar{y}^r)}{\alpha^{agr}} \right), \quad (10.109)$$

$$\frac{\partial a_r^{agr}(\bar{x}, \bar{y}^r)}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} = \frac{\partial}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} \text{avg}(I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)), \quad (10.110)$$

$$\frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial (1-v)} = \frac{\partial}{\partial (1-v)} \vec{H}^* \left(\frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{agr}, \dots, w_N^{agr}, p^{agr}, 1-v} \right), \quad (10.111)$$

$$\frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial p^{agr}} = \frac{\partial}{\partial p^{agr}} \vec{H}^* \left(\frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{agr}, \dots, w_N^{agr}, p^{agr}, 1-v} \right), \quad (10.112)$$

$$\frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial w_k^{agr}} = \frac{\partial}{\partial w_k^{agr}} \vec{H}^* \left(\frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{agr}, \dots, w_N^{agr}, p^{agr}, 1-v} \right), \quad (10.113)$$

$$\frac{\partial b_r^{agr}(\bar{x}, \bar{y}^r)}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} = \frac{\partial}{\partial I_{k,r}(\bar{x}, \bar{y}^r)} \vec{H}^* \left(\frac{I_{1,r}(\bar{x}, \bar{y}^r), \dots, I_{N,r}(\bar{x}, \bar{y}^r)}{w_1^{agr}, \dots, w_N^{agr}, p^{agr}, 1-v} \right). \quad (10.114)$$

Błędy propagowane przez blok wyostrzania określa się analogicznie jak w przypadku algorytmów uczenia dotyczących systemów nieelastycznych, które opisano w podrozdziałach 9.3–9.5.

Wyprowadzone powyżej algorytmy uczenia elastycznego systemu neuronowo-rozmytego wymagają wyznaczenia pochodnych dla różnego typu operatorów. Poniżej przedstawiono sposób obliczenia tych pochodnych.

10.6.1. Operatory podstawowe

Operator sumowania

$$y = \sum_{i=1}^n x_i \quad (10.115)$$

$$\frac{\partial y}{\partial x_i} = 1 \quad (10.116)$$

Operator mnożenia

$$y = \prod_{i=1}^n x_i \quad (10.117)$$

$$\frac{\partial y}{\partial x_i} = \prod_{\substack{j=1 \\ j \neq i}}^n x_j \quad (10.118)$$

Operator dzielenia

$$y = \frac{a}{b} \quad (10.119)$$

$$\frac{\partial y}{\partial a} = \frac{1}{b} \quad (10.120)$$

$$\frac{\partial y}{\partial b} = -\frac{a}{b^2} \quad (10.121)$$

Operator minimum

$$y = \min_{i=1 \dots n} \{x_i\} \quad (10.122)$$

$$\frac{\partial y}{\partial x_i} = \begin{cases} 1 & \text{dla } x_i = y \\ 0 & \text{dla } x_i \neq y \end{cases} \quad (10.123)$$

Operator maksimum

$$y = \max_{i=1 \dots n} \{x_i\} \quad (10.124)$$

$$\frac{\partial y}{\partial x_i} = \begin{cases} 1 & \text{dla } x_i = y \\ 0 & \text{dla } x_i \neq y \end{cases} \quad (10.125)$$

Operator kompromisowości

$$\tilde{N}_v(a) = (1 - f_z(v))(1 - a) + f_z(v)a \quad (10.126)$$

$$\frac{\partial \tilde{N}_v(a)}{\partial a} = 2f_z(v) - 1 \quad (10.127)$$

$$\frac{\partial \tilde{N}_v(a)}{\partial v} = (2a - 1) \frac{\partial f_z(v)}{\partial v} \quad (10.128)$$

Operator średniej arytmetycznej

$$\text{avg}(a_1, a_2, \dots, a_n) = \frac{1}{n} \sum_{i=1}^n a_i \quad (10.129)$$

$$\frac{\partial \text{avg}(a_1, a_2, \dots, a_n)}{\partial a_i} = \frac{1}{n} \quad (10.130)$$

Operator agregacji rozwiązań

$$G(a_1, a_2; \phi) = (1 - f_z(\phi))a_1 + f_z(\phi)a_2 \quad (10.131)$$

$$\frac{\partial G(a_1, a_2; \phi)}{\partial a_1} = 1 - f_z(\phi) \quad (10.132)$$

$$\frac{\partial G(a_1, a_2; \phi)}{\partial a_2} = f_z(\phi) \quad (10.133)$$

$$\frac{\partial G(a_1, a_2; \phi)}{\partial \phi} = -(a_1 - a_2) \frac{\partial f_z(\phi)}{\partial \phi} \quad (10.134)$$

Operator wyostrzania

$$\text{def}(a_1, a_2, \dots, a_n; w_1, w_2, \dots, w_n) = \text{def}(\mathbf{a}; \mathbf{w}) = \frac{\sum_{i=1}^n w_i a_i}{\sum_{i=1}^n a_i} \quad (10.135)$$

$$\frac{\partial \text{def}(\mathbf{a}; \mathbf{w})}{\partial a_j} = (w_j - \text{def}(\mathbf{a}; \mathbf{w})) \frac{1}{\sum_{i=1}^n a_i} \quad (10.136)$$

$$\frac{\partial \text{def}(\mathbf{a}; \mathbf{w})}{\partial w_j} = \left(a_j - \text{def}(\mathbf{a}; \mathbf{w}) \frac{\partial a_j}{\partial w_j} \right) \frac{1}{\sum_{i=1}^n a_i} \quad (10.137)$$

10.6.2. Funkcje przynależności

Funkcja przynależności Gaussa

$$\mu_A(x) = \exp \left(- \left(\frac{x - \bar{x}}{\sigma} \right)^2 \right) \quad (10.138)$$

$$\frac{\partial \mu_A(x)}{\partial x} = -\mu_A(x) \frac{2(x - \bar{x})}{\sigma^2} \quad (10.139)$$

$$\frac{\partial \mu_A(x)}{\partial \bar{x}} = \mu_A(x) \frac{2(x - \bar{x})}{\sigma^2} \quad (10.140)$$

$$\frac{\partial \mu_A(x)}{\partial \sigma} = \mu_A(x) \frac{2(x - \bar{x})^2}{\sigma^3} \quad (10.141)$$

Funkcja przynależności trójkątna

$$\mu_A(x) = \begin{cases} 0 & \text{dla } x \leq a \text{ lub } x \geq c \\ \frac{x - a}{b - a} & \text{dla } a \leq x \leq b \\ \frac{c - x}{c - b} & \text{dla } b \leq x \leq c \end{cases} \quad (10.142)$$

$$\frac{\partial \mu_A(x)}{\partial x} = \begin{cases} 0 & \text{dla } x < a \text{ lub } x > c \\ \frac{1}{2(b-a)} & \text{dla } x = a \\ \frac{1}{b-a} & \text{dla } a < x < b \\ \frac{c-2b+a}{2(c-b)(b-a)} & \text{dla } x = b \\ -\frac{1}{c-b} & \text{dla } b < x < c \\ -\frac{1}{2(c-b)} & \text{dla } x = c \end{cases} \quad (10.143)$$

$$\frac{\partial \mu_A(x)}{\partial a} = \begin{cases} 0 & \text{dla } x \leq a \text{ lub } x > b \\ \frac{1}{2(b-a)} & \text{dla } x = b \\ \frac{x-a}{(b-a)^2} & \text{dla } a \leq x < b \end{cases} \quad (10.144)$$

$$\frac{\partial \mu_A(x)}{\partial b} = \begin{cases} 0 & \text{dla } x \leq a \text{ lub } x \geq c \\ \frac{a-x}{(b-a)^2} & \text{dla } a \leq x < b \\ \frac{a-2b+c}{2(c-b)(b-a)} & \text{dla } x = b \\ \frac{c-x}{(c-b)^2} & \text{dla } b < x \leq c \end{cases} \quad (10.145)$$

$$\frac{\partial \mu_A(x)}{\partial c} = \begin{cases} 0 & \text{dla } x \leq b \text{ lub } x > c \\ \frac{1}{2(c-b)} & \text{dla } x = c \\ \frac{x-b}{(c-b)^2} & \text{dla } b < x < c \end{cases} \quad (10.146)$$

10.6.3. Funkcje zakresowe

Funkcja zakresowa dla parametrów $v \in [0, 1]$, $\lambda \in [0, 1]$, $\alpha^r \in [0, 1]$, $\alpha^l \in [0, 1]$, $\alpha^{agr} \in [0, 1]$, $w_{i,k}^r \in [0, 1]$, $i = 1, \dots, n$, $k = 1, \dots, N$, $w_k^{agr} \in [0, 1]$, $k = 1, \dots, N$

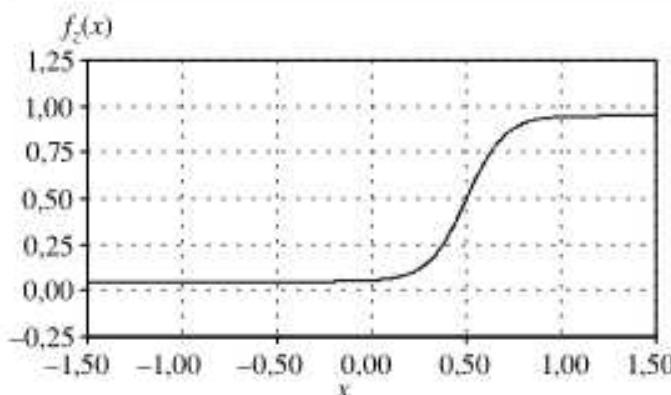
$$f_z(x) = \frac{p_{z3}}{1 + \exp(-(p_{z1}x - p_{z2}))} + p_{z4} \quad (10.147)$$

$$\frac{\partial f_z(x)}{\partial x} = -\frac{p_{z1}}{p_{z3}} (p_{z3} + p_{z4} - f_z(x))(p_{z4} - f_z(x)) \quad (10.148)$$

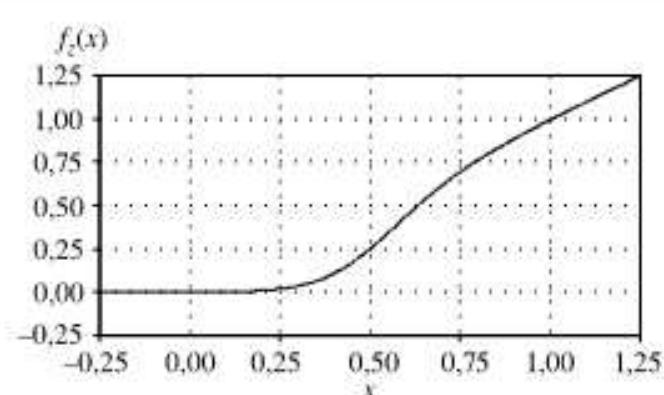
Funkcja zakresowa dla parametrów $p^r \in [0, \infty)$, $p^l \in [0, \infty)$, $p^{agr} \in [0, \infty)$

$$f_z(x) = \frac{x}{1 + \exp(-(p_{z1}x - p_{z2}))} + p_{z3} \quad (10.149)$$

$$\frac{\partial f_z(x)}{\partial x} = \frac{-p_{z3} + f_z(x)}{x} (1 + p_{z1}(p_{z3} + x - f_z(x))) \quad (10.150)$$



Rys. 10.18. Funkcja zakresowa dla $p_{z1} = 10$, $p_{z2} = 5$, $p_{z3} = 0,9$, $p_{z4} = 0,05$



Rys. 10.19. Funkcja zakresowa dla $p_{z1} = 10$, $p_{z2} = 5$, $p_{z3} = 0$

10.6.4. *H*-funkcje

Argument *H*-funkcji

$$\arg_i(a_i, w_i, v) = G\left(\frac{N(f_z(w_i)N(a_i)), f_z(w_i)a_i;}{v}\right) \quad (10.151)$$

$$\frac{\partial \arg_i(a_i, w_i, v)}{\partial a_i} = f_z(w_i) \quad (10.152)$$

$$\frac{\partial \arg_i(a_i, w_i, v)}{\partial w_i} = (a + v - 1) \frac{\partial f_z(w_i)}{\partial w_i} \quad (10.153)$$

$$\frac{\partial \arg_i(a_i, w_i, v)}{\partial v} = f_z(w_i) - 1 \quad (10.154)$$

H-funkcja Zadeha

$$H^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v(\max_{i=1, \dots, n} \{\tilde{N}_v(\arg_i(a_i, w_i, v))\}) \quad (10.155)$$

$$H^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v(h^*(\mathbf{a}; \mathbf{w}, v)) \quad (10.156)$$

gdzie

$$h^*(\mathbf{a}; \mathbf{w}, v) = \max_{i=1, \dots, n} \{\tilde{N}_v(\arg_i(a_i, w_i, v))\} \quad (10.157)$$

$$\frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial a_i} = \begin{cases} \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial a_i} & \text{dla } h^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v(\arg_i(a_i, w_i, v)) \\ 0 & \text{dla } h^*(\mathbf{a}; \mathbf{w}, v) \neq \tilde{N}_v(\arg_i(a_i, w_i, v)) \end{cases} \quad (10.158)$$

$$\frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial w_i} = \begin{cases} \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial w_i} & \text{dla } h^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v(\arg_i(a_i, w_i, v)) \\ 0 & \text{dla } h^*(\mathbf{a}; \mathbf{w}, v) \neq \tilde{N}_v(\arg_i(a_i, w_i, v)) \end{cases} \quad (10.159)$$

$$\begin{aligned} \frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial v} = & \frac{\partial f_z(v)}{\partial v} (2h^*(\mathbf{a}; \mathbf{w}, v) - 1) + \\ & + (2f_z(v) - 1) \max_{i=1, \dots, n} \left\{ \left(2f_z(v) - 1\right) \frac{\partial \arg_i(a_i, w_i, v)}{\partial v} + \right. \\ & \left. + (2\arg_i(a_i, w_i, v) - 1) \frac{\partial f_z(v)}{\partial v} \right\} \end{aligned} \quad (10.160)$$

***H*-funkcja algebraiczna**

$$H^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v \left(1 - \prod_{i=1}^n (1 - \tilde{N}_v(\arg_i(a_i, w_i, v))) \right) \quad (10.161)$$

$$H^*(\mathbf{a}; \mathbf{w}, v) = \tilde{N}_v(h^*(\mathbf{a}; \mathbf{w}, v)) \quad (10.162)$$

gdzie

$$h^*(\mathbf{a}; \mathbf{w}, v) = 1 - \prod_{i=1}^n (1 - \tilde{N}_v(\arg_i(a_i, w_i, v))) \quad (10.163)$$

$$\frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial a_i} = (2f_z(v) - 1)^2 \frac{\partial \arg_i(a_i, w_i, v)}{\partial a_i} \prod_{\substack{u=1 \\ u \neq i}}^n (1 - \tilde{N}_v(\arg_u(a_u, w_u, v))) \quad (10.164)$$

$$\frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial w_i} = (2f_z(v) - 1)^2 \frac{\partial \arg_i(a_i, w_i, v)}{\partial w_i} \prod_{\substack{u=1 \\ u \neq i}}^n (1 - \tilde{N}_v(\arg_u(a_u, w_u, v))) \quad (10.165)$$

$$\begin{aligned} \frac{\partial H^*(\mathbf{a}; \mathbf{w}, v)}{\partial v} = & (2h^*(\mathbf{a}; \mathbf{w}, v) - 1) \frac{\partial f_z(v)}{\partial v} + \\ & + (2f_z(v) - 1) \sum_{i=1}^n \left(\left(\begin{array}{l} (2\arg_i(a_i, w_i, v) - 1) \frac{\partial f_z(v)}{\partial v} + \\ + (2f_z(v) - 1) \frac{\partial \arg_i(a_i, w_i, v)}{\partial v} \end{array} \right) \times \right. \\ & \left. \times \prod_{\substack{u=1 \\ u \neq i}}^n (1 - \tilde{N}_v(\arg_u(a_u, w_u, v))) \right) \end{aligned} \quad (10.166)$$

***H*-funkcja Dombi**

$$\tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v) = \tilde{N}_v \left(1 - \left(1 + \left(\sum_{i=1}^n (\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{-f_{z1}(p)} \right)^{\frac{1}{f_{z1}(p)}} \right)^{-1} \right) \quad (10.167)$$

$$\tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v) = \tilde{N}_v(1 - \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)) \quad (10.168)$$

$$p \in (0, \infty) \quad (10.169)$$

gdzie

$$\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) = \left(1 + \left(\sum_{i=1}^n (\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{-f_{z1}(p)} \right)^{\frac{1}{f_{z1}(p)}} \right)^{-1} \quad (10.170)$$

$$\begin{aligned} \frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial a_i} = & (2f_z(v) - 1)^2 \frac{(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)^{1-f_{z1}(p)}}{\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-2}} \times \\ & \times \frac{(\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{-f_{z1}(p)-1}}{\tilde{N}_v(\arg_i(a_i, w_i, v))^2} \times \\ & \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial a_i} \end{aligned} \quad (10.171)$$

$$\begin{aligned} \frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial w_i} &= (2f_z(v) - 1)^2 \frac{(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)^{1-f_{z1}(p)}}{\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-2}} \times \\ &\quad \times \frac{(\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{-f_{z1}(p)-1}}{\tilde{N}_v(\arg_i(a_i, w_i, v))^2} \times \\ &\quad \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial w_i} \end{aligned} \quad (10.172)$$

$$\begin{aligned} \frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial p} &= \frac{2f_z(v) - 1}{f_{z1}(p)} \frac{(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)^{1-f_{z1}(p)}}{\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-2}} \times \\ &\quad \times \left(\sum_{i=1}^n \frac{-\ln(\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)}{(\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{f_{z1}(p)}} \right) \times \\ &\quad \times \frac{-\ln(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)}{(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)^{f_{z1}(p)}} \\ &\quad \times \frac{\partial f_{z1}(p)}{\partial p} \end{aligned} \quad (10.173)$$

$$\begin{aligned} \frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial v} &= (1 - 2\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)) \frac{\partial f_z(v)}{\partial v} + \\ &\quad + (2f_z(v) - 1) \frac{(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-1} - 1)^{1-f_{z1}(p)}}{\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{-2}} \times \\ &\quad \times \sum_{i=1}^n \left(\frac{(\tilde{N}_v(\arg_i(a_i, w_i, v))^{-1} - 1)^{-f_{z1}(p)-1}}{\tilde{N}_v(\arg_i(a_i, w_i, v))^2} \times \right. \\ &\quad \left. \times \left((2f_z(v) - 1) \frac{\partial \arg_i(a_i, w_i, v)}{\partial v} + \right. \right. \\ &\quad \left. \left. + (2\arg_i(a_i, w_i, v) - 1) \frac{\partial f_z(v)}{\partial v} \right) \right) \end{aligned} \quad (10.174)$$

H-funkcja Yagera

$$\tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v) = \tilde{N}_v \left(\min \left\{ 1, \left(\sum_{i=1}^n \tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)} \right)^{\frac{1}{f_{z1}(p)}} \right\} \right) \quad (10.175)$$

$$\tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v) = \tilde{N}_v \left(\min \{ 1, \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \} \right) \quad (10.176)$$

$$\tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v) = \begin{cases} \tilde{N}_v(\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)) & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \leq 1 \\ \tilde{N}_v(1) & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) > 1 \end{cases} \quad (10.177)$$

$$p \in (0, \infty) \quad (10.178)$$

gdzie

$$\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) = \left(\sum_{i=1}^n \tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)} \right)^{\frac{1}{f_{z1}(p)}} \quad (10.179)$$

$$\frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial a_i} = \begin{cases} (2f_z(v) - 1)^2 \times \\ \times \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{1-f_{z1}(p)} \times \\ \times \tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)-1} \times & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \leq 1 \\ \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial a_i} \\ 0 & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) > 1 \end{cases} \quad (10.180)$$

$$\frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial w_i} = \begin{cases} (2f_z(v) - 1)^2 \times \\ \times \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{1-f_{z1}(p)} \times \\ \times \tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)-1} \times & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \leq 1 \\ \times \frac{\partial \arg_i(a_i, w_i, v)}{\partial w_i} \\ 0 & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) > 1 \end{cases} \quad (10.181)$$

$$\frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial p} = \begin{cases} \frac{2f_z(v) - 1}{f_{z1}(p)} \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{1-f_{z1}(p)} \times \\ \times \left(\sum_{i=1}^n \frac{\ln(\tilde{N}_v(\arg_i(a_i, w_i, v)))}{\tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)}} + \right) \frac{\partial f_{z1}(p)}{\partial p} & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \leq 1 \\ 0 & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) > 1 \end{cases} \quad (10.182)$$

$$\frac{\partial \tilde{H}^*(\mathbf{a}; \mathbf{w}, p, v)}{\partial v} = \begin{cases} (2\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) - 1) \frac{\partial f_z(v)}{\partial v} + \\ + (2f_z(v) - 1) \frac{1}{\tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v)^{f_{z1}(p)-1}} \times \\ \times \sum_{i=1}^n \left(\tilde{N}_v(\arg_i(a_i, w_i, v))^{f_{z1}(p)-1} \times \right. \\ \left. \times \left((2f_z(v) - 1) \frac{\partial \arg_i(a_i, w_i, v)}{\partial v} + \right) \right) & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) \leq 1 \\ \frac{\partial f_z(v)}{\partial v} & \text{dla } \tilde{h}^*(\mathbf{a}; \mathbf{w}, p, v) > 1 \end{cases} \quad (10.183)$$

10.7. Przykłady symulacyjne

Przedstawimy wyniki symulacji dla wcześniej opisanych elastycznych systemów neuronowo-rozmytych. Symulacje dotyczą problemów polimeryzacji, modelowania smaku ryżu, klasyfikacji kwiatów irysa oraz klasyfikacji wina przedstawionych w podrozdziale 9.2. W celu przypomnienia problemy te zestawiono w tabeli 10.2. Dla każdego przykładu symulacyjnego przeprowadzono dwie serie symulacji. Każda seria została przeprowadzona i opisana w analogiczny sposób:

- W pierwszym eksperymencie uczone były tylko parametry funkcji przynależności wejściowych i wyjściowych zbiorów rozmytych oraz parametr modelu wnioskowania $v \in [0, 1]$. Wartość tego parametru po zakończeniu procesu uczenia należy do zbioru $v \in \{0, 1\}$.

- W drugim eksperymencie również uczone były parametry funkcji przynależności wejściowych i wyjściowych zbiorów rozmytych, natomiast wartość parametru modelu wnioskowania v została przyjęta jako przeciwna (0 lub 1) do uzyskanej w eksperymencie pierwszym. Jak zobaczymy, dokładność uzyskiwana w tym eksperymencie jest gorsza od uzyskiwanej w eksperymencie pierwszym.

- W eksperymencie trzecim uczone były takie parametry jak w eksperymencie pierwszym, a ponadto parametry elastyczności $\alpha^r \in [0, 1]$, $\alpha^l \in [0, 1]$, $\alpha^{agr} \in [0, 1]$ oraz parametry kształtu zastosowanych operatorów $p^r \in [0, \infty)$, $p^l \in [0, \infty)$, $p^{agr} \in [0, \infty)$. Ostatnie z wymienionych parametrów występują w przypadku stosowania nastawnych H -funkcji typu Dombi i Yagera (w drugiej serii eksperymentów).

- W eksperymencie czwartym uczone były takie parametry jak w eksperymencie trzecim, a ponadto wagi przesłanek reguł $w_{i,k}^r \in [0, 1]$, $i = 1, \dots, n$, $k = 1, \dots, N$, oraz wagi poszczególnych reguł $w_k^{agr} \in [0, 1]$, $k = 1, \dots, N$. Wartości wag po zakończeniu procesu uczenia ilustrują diagramy, na których wagi przesłanek i wagi reguł oddzielono pionową przerywaną linią. Przy sporządzaniu diagramów przyjęto zasadę, że im bardziej zaciemnione pole symbolizujące daną wagę, tym wartość wagi jest bliższa zera.

W pierwszej serii symulacji, w każdym z czterech opisanych wyżej eksperymentów, wykorzystane były nienastawne H -funkcje oraz H -implikacje Zadeha i algebraiczne. W drugiej serii eksperymentów zamiast operatorów nienastawnych zastosowano nastawne H -funkcje oraz H -implikacje Dombi i Yagera.

Tabela 10.2. Wykorzystane przykłady symulacyjne

Problem symulacyjny	Rodzaj problemu	Liczba wejść	Długość ciągu uczącego	Długość ciągu testowego
Polimeryzacja	aproksymacja	3	70	—
Modelowanie smaku ryżu	aproksymacja	5	75	30
Klasyfikacja kwiatów irysa	klasyfikacja	4	105	45
Klasyfikacja wina	klasyfikacja	13	125	53

10.7.1. Polimeryzacja

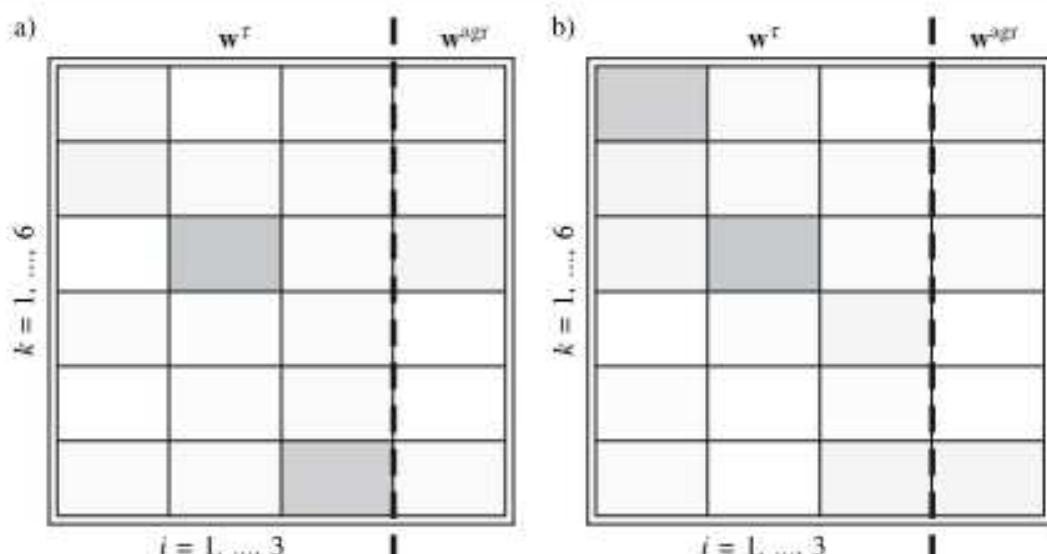
Wyniki symulacji dla problemu Polimeryzacja zamieszczono w tabeli 10.3 dla H -funkcji nienastawnych (Zadeha i algebraicznych) i w tabeli 10.4 dla H -funkcji nastawnych (Dombi i Yagera). Ponadto, dla eksperymentu (iv) wartości wag przesłanek reguł $w_{i,k}^r \in [0, 1]$ i wartości wag reguł $w_k^{agr} \in [0, 1]$ rozważanych systemów z H -funkcjami

Tabela 10.3. Wyniki symulacji systemu elastycznego z nieparametryzowanymi H -funkcjami — problem polimeryzacji

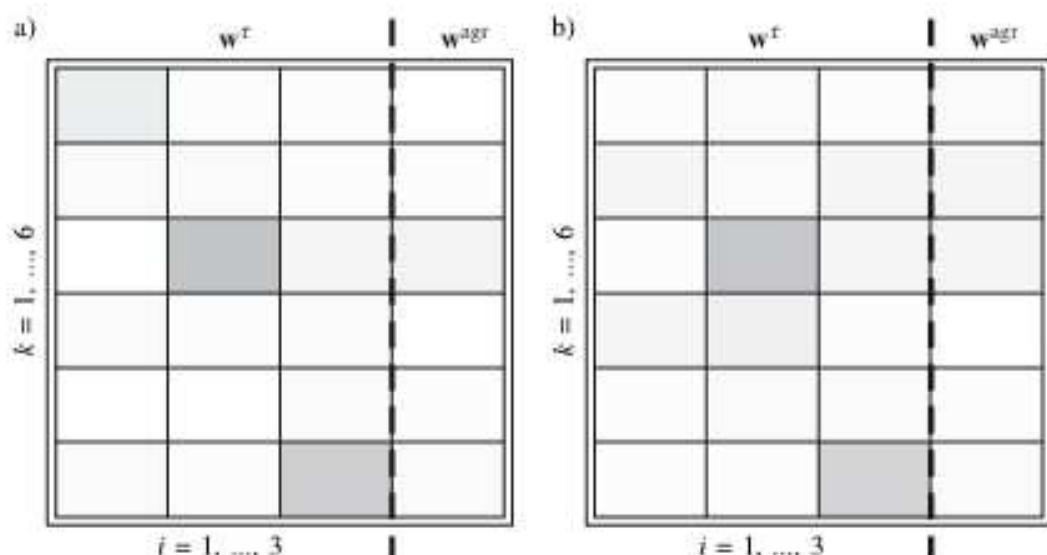
SYSTEM ELASTYCZNY Z NIEPARAMETRYZOWANYMI H -FUNKCJAMI (POLIMERYZACJA)						
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		RMSE (ciąg uczący)	
			H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.
i	v	0,5	0,0000	0,0000	0,0096	0,0060
ii	v	1	—	—	0,0115	0,0063
iii	v	0,5	0,0000	0,0000		
	α^r	1	0,7158	0,9678		
	α^l	1	0,7613	0,9992	0,0059	0,0056
	α^{agr}	1	0,7277	0,9930		
iv	v	0,5	0,0000	0,0000		
	α^r	1	0,6941	0,9987		
	α^l	1	0,7783	0,9992		
	α^{agr}	1	0,6713	0,9334	0,0056	0,0044
	w^r	1	rys.10.20a	rys.10.20b		
	w^{agr}	1	rys.10.20a	rys.10.20b		

Tabela 10.4. Wyniki symulacji systemu elastycznego z parametryzowanymi H -funkcjami — problem polimeryzacji

SYSTEM ELASTYCZNY Z PARAMETRYZOWANYMI H -FUNKCJAMI (POLIMERYZACJA)						
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		RMSE (ciąg uczący)	
			H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera
i	v	0,5	0,0000	0,0000	0,0117	0,0110
ii	v	1	—	—	0,0133	0,0113
iii	v	0,5	0,0000	0,0000		
	p^r	10	9,9714	10,2089		
	p^l	10	10,0042	10,2594		
	p^{agr}	10	9,9835	9,3991	0,0077	0,0061
	α^r	1	0,6996	0,1624		
	α^l	1	0,7743	0,5344		
	α^{agr}	1	0,9941	0,9942		
iv	v	0,5	0,0000	0,0000		
	p^r	10	13,1310	7,5714		
	p^l	10	15,3619	11,7834		
	p^{agr}	10	3,4720	13,9273		
	α^r	1	0,7127	0,1375	0,0069	0,0053
	α^l	1	0,7148	0,4742		
	α^{agr}	1	0,9335	0,9910		
	w^r	1	rys.10.21a	rys.10.21b		
	w^{agr}	1	rys.10.21a	rys.10.21b		



Rys. 10.20. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem polimeryzacji w przypadku a) H -funkcji Zadeha, b) H -funkcji algebraicznej

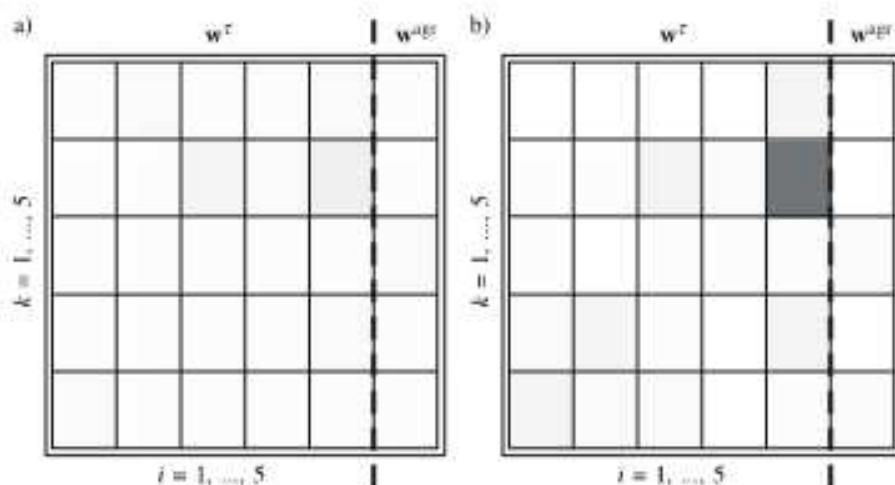


Rys. 10.21. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem polimeryzacji w przypadku a) H -funkcji Dombi, b) H -funkcji Yagera

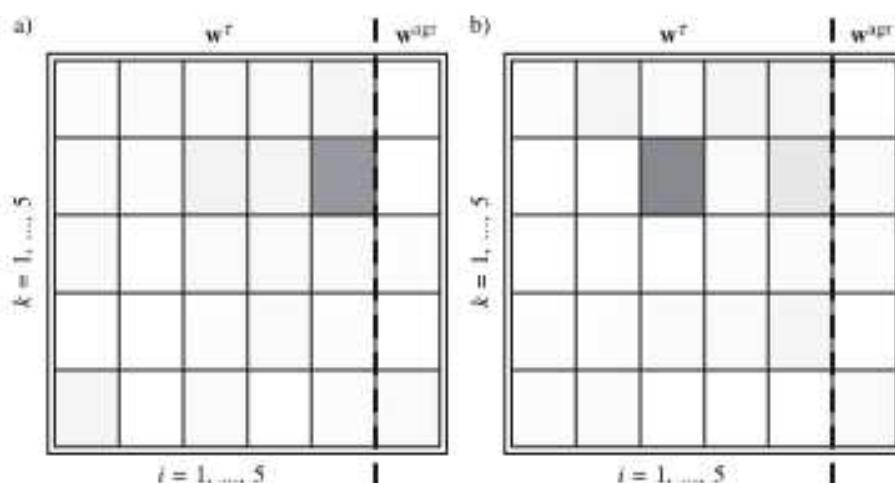
nienastawnymi symbolicznie pokazano na rysunku 10.20, natomiast wartości wag systemów z H -funkcjami nastawnymi ilustruje rysunek 10.21.

10.7.2. Modelowanie smaku ryżu

Rezultaty symulacji dla problemu modelowania smaku ryżu zamieszczono w tabeli 10.5 dla H -funkcji nienastawnych (Zadeha i algebraicznych) i w tabeli 10.6 dla H -funkcji nastawnych (Dombi i Yagera). Ponadto, dla eksperymentu (iv) wartości wag przesłanek reguł $w_{i,k}^r \in [0, 1]$ i wartości wag reguł $w_k^{agr} \in [0, 1]$ rozważanych systemów z H -funkcjami nienastawnymi symbolicznie pokazano na rysunku 10.22, natomiast wartości wag systemów z H -funkcjami nastawnymi ilustruje rysunek 10.23.



Rys. 10.22. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem modelowania smaku ryżu w przypadku a) H -funkcji Zadeha, b) H -funkcji algebraicznej



Rys. 10.23. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem modelowania smaku ryżu w przypadku a) H -funkcji Dombi, b) H -funkcji Yagera

Tabela 10.5. Wyniki symulacji systemu elastycznego z nieparametryzowanymi H -funkcjami — problem modelowania smaku ryżu

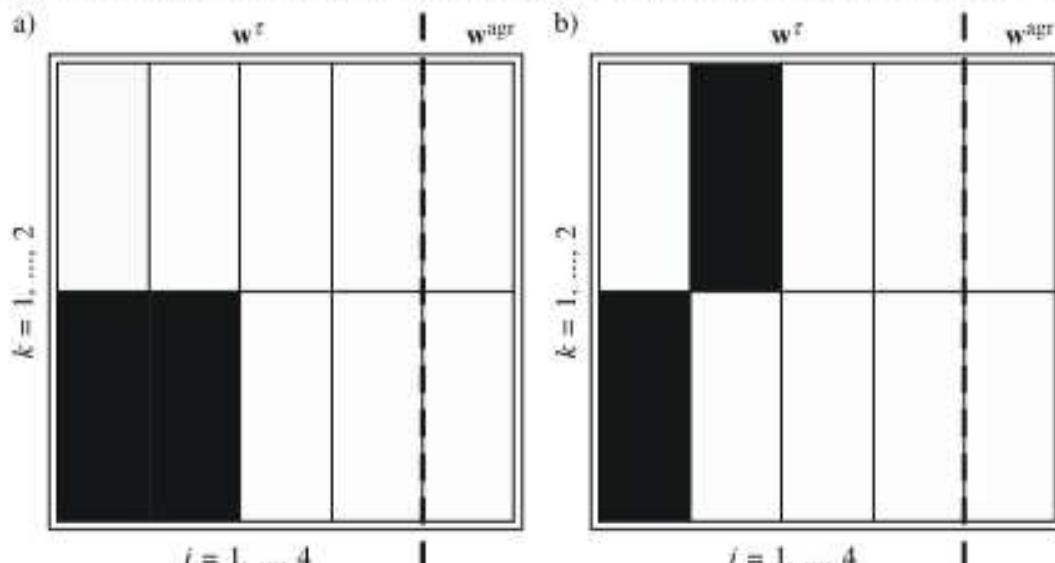
SYSTEM ELASTYCZNY Z NIEPARAMETRYZOWANYMI H -FUNKcjAMI (MODELOWANIE SMAKU RYŻU)						
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		RMSE (ciąg uczący)	
			H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.
i	ν	0,5	0,0000	0,0000	0,0184	0,0185
ii	ν	1	—	—	0,0186	0,0192
iii	ν α^z α^I α^{agr}	0,5 1 1 1	0,0000 0,2954 0,9843 0,4658	0,0000 0,9972 0,9979 0,9958	0,0163	0,0173
iv	ν α^z α^I α^{agr} w^z w^{agr}	0,5 1 1 1 1 1	0,0000 0,3101 0,9575 0,5496 rys.10.22a rys.10.22a	0,0000 0,9519 0,9512 0,9085 rys.10.22b rys.10.22b	0,0140	0,0159

Tabela 10.6. Wyniki symulacji systemu elastycznego z parametryzowanymi H -funkcjami — problem modelowania smaku ryżu

SYSTEM ELASTYCZNY Z PARAMETRYZOWANYMI H -FUNKcjAMI (MODELOWANIE SMAKU RYŻU)						
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		RMSE (ciąg uczący)	
			H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera
i	v	0,5	0,0000	0,0000	0,0186	0,0187
ii	v	1	—	—	0,0192	0,0197
iii	v	0,5	0,0000	0,0000		
	p^r	10	9,9268	10,7365		
	p^l	10	10,0026	10,1154		
	p^{agr}	10	9,7692	10,8200	0,0181	0,0184
	α^r	1	0,4606	0,6895		
	α^l	1	0,9943	0,9993		
	α^{agr}	1	0,9865	0,9728		
iv	v	0,5	0,0000	0,0000		
	p^r	10	10,1449	10,9117		
	p^l	10	9,9448	10,0472		
	p^{agr}	10	9,2063	10,0148		
	α^r	1	0,4380	0,6763	0,0160	0,0169
	α^l	1	0,9201	0,9263		
	α^{agr}	1	0,8967	0,9927		
	w^r	1	rys.10.23a	rys.10.23b		
	w^{agr}	1	rys.10.23a	rys.10.23b		

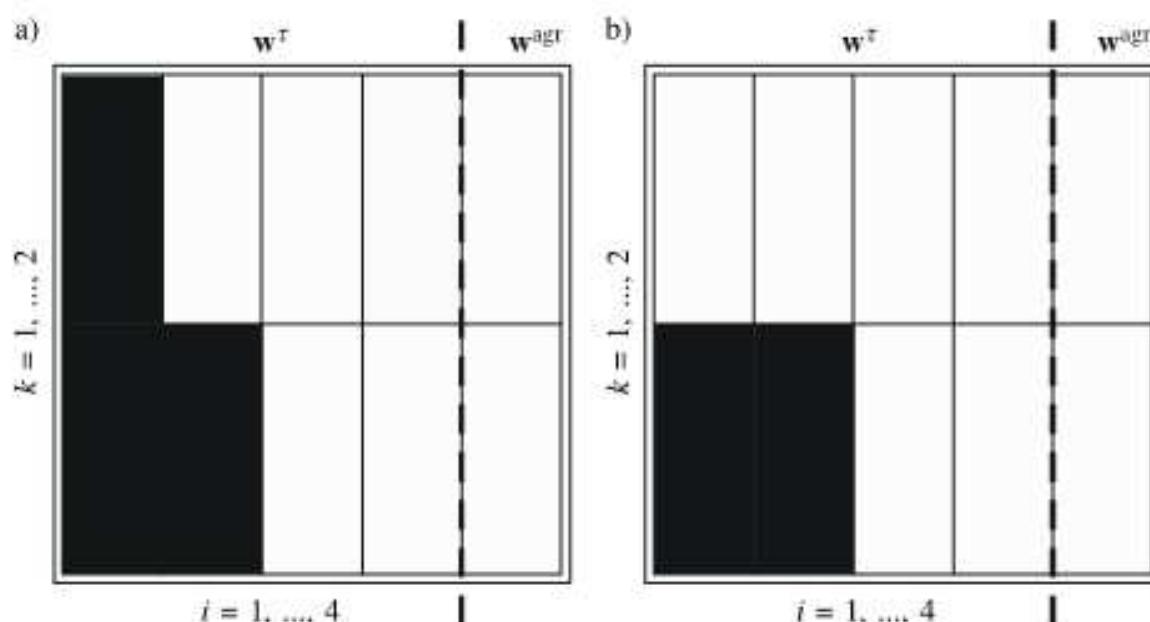
10.7.3. Klasyfikacja kwiatu irysa

Wyniki symulacji dla problemu klasyfikacji kwiatu irysa zamieszczone w tabeli 10.7 dla H -funkcji nienastawnych (Zadeha i algebraicznych) i w tabeli 10.8 dla H -funkcji



Rys. 10.24. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem klasyfikacji kwiatów irysa w przypadku a) H -funkcji Zadeha, b) H -funkcji algebraicznej

nastawnych (Dombi i Yagera). Ponadto, dla eksperymentu (iv) wartości wag przesłanek reguł $w_{i,k}^r \in [0, 1]$ i wartości wag reguł $w_k^{agr} \in [0, 1]$ rozważanych systemów z H -funkcjami nienastawnymi symbolicznie pokazano na rysunku 10.24, natomiast wartości wagi systemów z H -funkcjami nastawnymi ilustruje rysunek 10.25.



Rys. 10.25. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązywającego problem klasyfikacji kwiatu irysa w przypadku a) H -funkcji Dombi, b) H -funkcji Yagera

Tabela 10.7. Wyniki symulacji systemu elastycznego z nieparametryzowanymi H -funkcjami — problem klasyfikacji kwiatu irysa

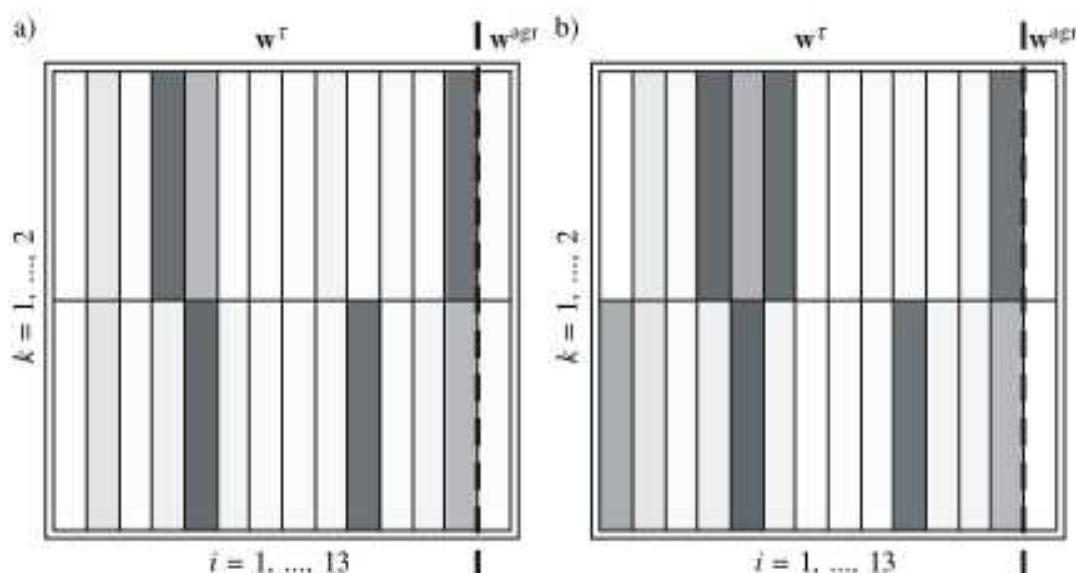
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		Liczba błędów [%](ciąg uczący)		Liczba błędów [%](ciąg testowy)	
			H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.
i	ν	0,5	1,0000	1,0000	0,95	0,95	4,44	4,44
ii	ν	0	—	—	0,95	0,95	6,67	6,67
iii	ν	0,5	1,0000	1,0000				
	α^r	1	0,2032	0,9922				
	α^l	1	0,9891	0,6082	0,00	0,95	4,44	4,44
	α^{agr}	1	0,9994	0,9998				
iv	ν	0,5	1,0000	1,0000				
	α^r	1	0,2442	0,9592				
	α^l	1	0,9845	0,5753				
	α^{agr}	1	0,9650	0,9937	0,00	0,00	4,44	4,44
	w^r	1	rys.10.24a	rys.10.24b				
	w^{agr}	1	rys.10.24a	rys.10.24b				

Tabela 10.8. Wyniki symulacji systemu elastycznego z parametryzowanymi H -funkcjami — problem klasyfikacji kwiatu irysa

SYSTEM ELASTYCZNY Z PARAMETRYZOWANYMI H -FUNKCJAMI (KLASYFIKACJA KWIATU IRYSA)								
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		Liczba błędów [%](ciąg uczący)		Liczba błędów [%](ciąg testowy)	
			H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera
i	v	0,5	1,0000	1,0000	0,00	0,95	4,44	4,44
ii	v	0	—	—	0,95	0,95	6,67	6,67
iii	v	0,5	1,0000	1,0000				
	p^r	10	13,2031	4,3306				
	p^l	10	10,0001	7,5741				
	p^{agr}	10	9,9974	10,1209	0,00	0,00	4,44	4,44
	α^r	1	0,8259	0,7846				
	α^l	1	0,9924	0,9931				
	α^{agr}	1	0,9985	0,9985				
iv	v	0,5	1,0000	1,0000				
	p^r	10	13,5253	4,3621				
	p^l	10	10,8610	8,0120				
	p^{agr}	10	9,4218	9,3590				
	α^r	1	0,8739	0,8068	0,00	0,00	2,22	2,22
	α^l	1	0,9871	0,9731				
	α^{agr}	1	0,9698	0,9661				
	w^r	1	rys.10.25a	rys.10.25b				
	w^{agr}	1	rys.10.25a	rys.10.25b				

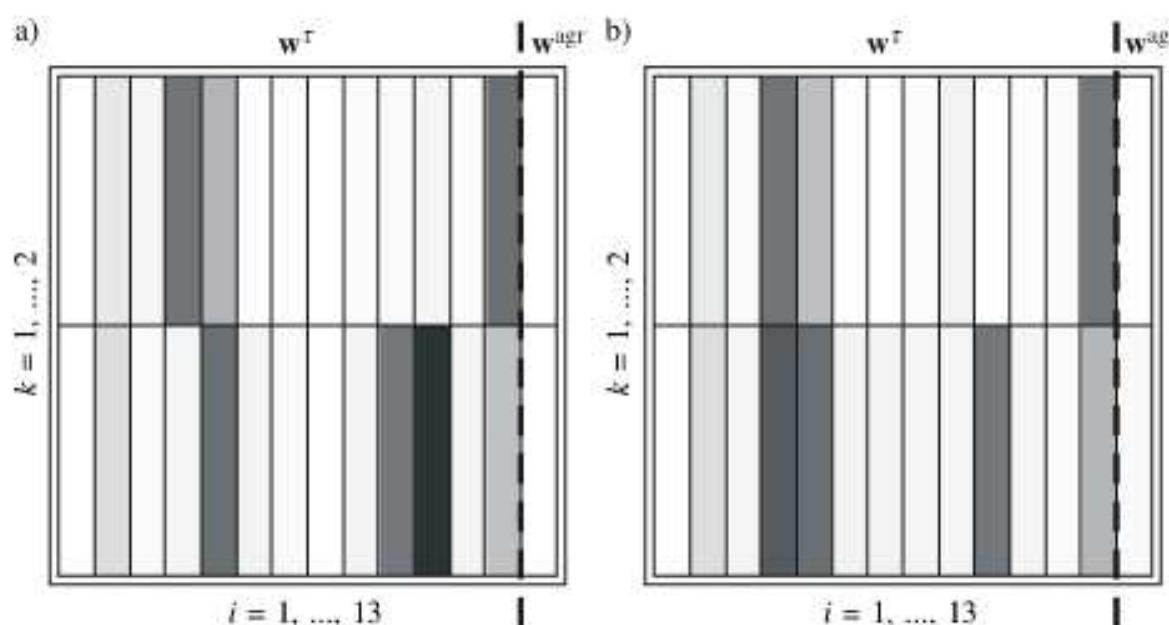
10.7.4. Rozpoznawanie gatunku wina

Rezultaty symulacji dla problemu klasyfikacji wina zamieszczone w tabeli 10.9 dla H -funkcji nienastawnych (Zadeha i algebraicznych) i w tabeli 10.10 dla H -funkcji



Rys. 10.26. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem rozpoznawania gatunku wina w przypadku a) H -funkcji Zadeha, b) H -funkcji algebraicznej

nastawnych (Dombi i Yagera). Ponadto, dla eksperymentu (iv) wartości wag przesłanek reguł $w_{i,k}^r \in [0, 1]$ i wartości wag reguł $w_k^{\text{agr}} \in [0, 1]$ rozważanych systemów z H -funkcjami nienastawnymi symbolicznie pokazano na rysunku 10.26, natomiast wartości wag systemów z H -funkcjami nastawnymi ilustruje rysunek 10.27.



Rys. 10.27. Wagi przesłanek reguł i wagi reguł dla systemu elastycznego rozwiązującego problem rozpoznawania gatunku wina w przypadku a) H -funkcji Dombi, b) H -funkcji Yagera

Tabela 10.9. Wyniki symulacji systemu elastycznego z nieparametryzowanymi H -funkcjami — problem rozpoznawania gatunku wina

SYSTEM ELASTYCZNY Z NIEPARAMETRYZOWANYMI H -FUNKCJAMI (ROZPOZNAWANIE GATUNKU WINA)								
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		Liczba błędów [%](ciąg uczący)		Liczba błędów [%](ciąg testowy)	
			H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.	H -funkcja Zadeha	H -funkcja algebr.
i	v	0,5	1,0000	1,0000	0,00	0,00	3,77	1,89
ii	v	0	—	—	0,80	0,80	3,77	3,77
iii	v	0,5	1,0000	1,0000				
	α^r	1	0,0004	0,0036				
	α^l	1	0,9907	0,9986	0,00	0,00	1,89	1,89
	α^{agr}	1	0,9938	0,9908				
iv	v	0,5	1,0000	1,0000				
	α^r	1	0,0329	0,0180				
	α^l	1	0,9987	0,9756				
	α^{agr}	1	0,9896	0,9861	0,00	0,00	0,00	0,00
	w^r	1	rys.10.26a	rys.10.26b				
	w^{agl}	1	rys.10.26a	rys.10.26b				

Tabela 10.10. Wyniki symulacji systemu elastycznego z parametryzowanymi H -funkcjami — problem rozpoznawania gatunku wina

SYSTEM ELASTYCZNY Z PARAMETRYZOWANYMI H -FUNKcjAMI (ROZPOZNAWANIE GATUNKU WINA)								
Numer symulacji	Nazwa parametru elastyczności	Wartość początkowa	Wartość końcowa po uczeniu		Liczba błędów [%](ciąg uczący)		Liczba błędów [%](ciąg testowy)	
			H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera	H -funkcja Dombi	H -funkcja Yagera
i	ν	0,5	1,0000	1,0000	0,00	0,00	1,89	1,89
ii	ν	0	—	—	0,00	0,00	3,77	3,77
iii	ν	0,5	1,0000	1,0000				
	p^r	10	9,9999	10,0498				
	p^l	10	10,0005	9,9936				
	p^{agr}	10	9,9991	10,0014	0,00	0,00	1,89	1,89
	α^r	1	0,0032	0,0029				
	α^l	1	0,9911	0,9917				
	α^{agr}	1	0,9919	0,9920				
iv	ν	0,5	1,0000	1,0000				
	p^r	10	7,8330	6,9528				
	p^l	10	11,7084	13,3122				
	p^{agr}	10	14,3699	12,1427				
	α^r	1	0,0028	0,0389	0,00	0,00	0,00	0,00
	α^l	1	0,9826	0,9740				
	α^{agr}	1	0,9914	0,9599				
	w^r	1	rys.10.27a	rys.10.27b				
	w^{agr}	1	rys.10.27a	rys.10.27b				

10.8. Uwagi

Przedstawiona w tym rozdziale koncepcja elastycznych systemów neuronowo-rozmytych pozwala wyznaczyć typ systemu (Mamdaniego lub logiczny) w wyniku procesu uczenia. Z przykładów symulacyjnych podanych w podrozdziale 10.7 wynika, że system elastyczny po zakończeniu procesu uczenia staje się systemem Mamdaniego (parametr $\nu = 0$) dla problemów aproksymacji lub identyfikacji. Natomiast dla problemów klasyfikacji system elastyczny w wyniku uczenia staje się systemem typu logicznego (parametr $\nu = 1$). Rezultaty te można traktować jako rekomendację systemu typu Mamdaniego do rozwiązywania problemów aproksymacji lub identyfikacji oraz systemu logicznego do rozwiązania problemów klasyfikacji. Należy wspomnieć, że koncepcję miękkich norm trójkątnych podali Yager i Filev [262], natomiast Klement [111] i Lowen [128] szczegółowo przedstawili różne typy parametryzowanych norm trójkątnych. Różne typy elastycznych struktur neuronowo-rozmytych zaproponował Cpałka [30]. Szerzej tematykę tych systemów porusza monografia [225]. Zainteresowanego czytelnika odsyłamy również do prac [210–212, 215, 217, 218, 220, 223, 227].

LITERATURA

- [1] Aliev R.A., Aliev R.R., *Soft computing and its applications*, World Scientific, Singapore 2001.
- [2] Arabas J., *Wykłady z algorytmów ewolucyjnych*, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- [3] Babuška R., *Fuzzy modeling for control*, Kluwer Academic Publishers, Boston 1998.
- [4] Backer E., *Computer-Assisted Reasoning in Cluster Analysis*, Prentice Hall, New York 1995.
- [5] Badźmirowski K., Kubiś M., *Systemy ekspertowe*, Przemysłowy Instytut Elektroniki, Warszawa 1991.
- [6] Bartlett P., Downs T., *Training a neural networks with a genetic algorithm*, Technical Report, Dept. of Elec. Eng., Univ. of Queensland, 1990.
- [7] Belew R.K., McInerney J., Schraudolph N.N., *Evolving networks: Using genetic algorithms with connectionist learning*, CSE technical report CS90-174, La Jolla, CA: University of California at San Diego 1990.
- [8] Bellman R.E., Giertz M., On the analytical formalism of fuzzy sets, *Information Sciences*, **5**, 149–156 (1975).
- [9] Bezdek J.C., Pal S.K., *Fuzzy Models for Pattern Recognition*, IEEE Press, New York 1992.
- [10] Bezdek J., Keller J., Krisnapuram R., Pal N.R., *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Press, 1999.
- [11] Bilski J., Rutkowski L., A fast training algorithm for neural networks, *IEEE Trans. on Circuits and Systems II*, June 1998, 749–753 (1998).
- [12] Bishop Ch.M., *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, New York 1995.
- [13] Bojadziev G., Bojadziev M., *Fuzzy Logic for Business, Finance and Management*, World Scientific, Singapore 1999.

- [14] Bonabeau E., Theraulaz G., Mądrości roju, *Świat nauki*, Nr 6, 49–56, Czerwiec 2000.
- [15] Branke J., *Evolutionary Algorithm for Neural Network Design and Training*, University of Karlsruhe, 1995.
- [16] Brindle M., *Genetic Algorithms for Function Optimization*, Ph.D. dissertation, University of Alberta, 1981.
- [17] Brown M., Harris C., *Neuro-fuzzy Adaptive Modelling and Control*, Prentice Hall PTR, Upper Saddle River, NJ 1994.
- [18] Bubnicki Z., *Teoria i algorytmy sterowania*, Wydawnictwo Naukowe PWN, Warszawa 2002.
- [19] Bubnicki Z., *Analysis and Decision Making in Uncertain Systems*, Series: Communications and Control Engineering, Springer-Verlag, Heidelberg 2004.
- [20] Cacoullos T., Estimation of a multivariate density, *Ann. Inst. Statist. Math.*, **18**, 179–189 (1965).
- [21] Calvo T., Mayor G., Mesiar R. (red.), *Aggregation Operators. New Trends and Applications*, Physica-Verlag. A Springer-Verlag Company, New York 2002.
- [22] Capcarrère M., Tettamazi A., Tomassini M., Sipper M., A statistical study of a class of cellular evolutionary algorithms, *Evolutionary Computation*, **7**(3), 255–274 (1999).
- [23] Chi Z., Yan H., Pham T., *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*, World Scientific, Singapore 1996.
- [24] Chong E.K.P., Żak S.H., *An Introduction to Optimization*, John Wiley, 1996.
- [25] Chromiec J., Strzemieczna E., *Sztuczna inteligencja. Metody konstrukcji i analizy systemów eksperckich*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994.
- [26] Cichocki A., Unbehauen R., *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons — Interscience, New York 1993.
- [27] Cichosz P., *Systemy uczące się*, Wydawnictwa Naukowo-Techniczne, Warszawa 2000.
- [28] Cordón O., Herrera F., Evolutionary design of TSK fuzzy ruled based systems using (μ, λ) -evolutionary strategies, *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, vol. 1, 509–514, Barcelona 1997.
- [29] Cordón O., Herrera F., Hoffman F., Magdalena L., *Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific, Singapore 2001.
- [30] Cpałka K., *Elastyczne neuronowo-rozmyte systemy wnioskujące*, Praca doktorska, Politechnika Częstochowska, Częstochowa 2001.
- [31] Cytowski J., *Algorytmy genetyczne*, Akademicka Oficyna Wydawnicza, Warszawa 1996.

- [32] Czabański R., *Automatyczne wyznaczanie reguł rozmytych JEŻELI-TO na podstawie danych numerycznych*, Praca doktorska, Politechnika Śląska, Gliwice 2002.
- [33] Czogała E., Pedrycz W., *Elementy i metody teorii zbiorów rozmytych*, PWN, Warszawa 1985.
- [34] Czogała E., Łęski J., *Fuzzy and Neuro-Fuzzy Intelligent Systems*, Physica-Verlag, Heidelberg, New York 2000.
- [35] Davis L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York 1991.
- [36] Dąbrowski M., Rudeński A., Zastosowanie procedur ewolucyjnych w optymalizacji maszyn elektrycznych, *X Konferencja Naukowo-Techniczna: Zastosowania komputerów w elektrotechnice*, 11–12, Poznań/Kiekrz 2005.
- [37] Dąbrowski M., Rudeński A., Dyskretne zmienne niezależne w niedeterministycznej optymalizacji maszyny elektrycznej, *Proc. of XLI International Symposium on Electrical Machines SME-2005*, Opole–Jarnołtówek 2005.
- [38] De Jong K.A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Dissertation, University of Michigan, 1975.
- [39] De Silva C.W., *Intelligent Control: Fuzzy Logic Applications*, CRC Press, Boca Raton 1995.
- [40] Driankov D., Hellendoorn H., Reinfrank M., *Wprowadzenie do sterowania rozmytego*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
- [41] Dubois D., Prade H., Operations on fuzzy numbers, *Intern. Journal System Science*, **9**, 613–626 (1978).
- [42] Dubois D., Prade H., *Fuzzy Sets and Systems: Theory and Applications, Mathematics in Science Engineering*, Academic Press, Inc., vol. 144, San Diego 1980.
- [43] Dubois D., Prade H., Rough fuzzy sets and fuzzy rough sets, *International J. General Systems*, **17**(2–3), 191–209 (1990).
- [44] Dubois D., Prade H., Putting Rough Sets and Fuzzy Sets Together, w: Słowiński R. (red.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publishers, 203–232, Dordrecht 1992.
- [45] Duch W., *Fascynujący świat programów komputerowych*, Wydawnictwo NAKOM, Poznań 1997.
- [46] Duch W., Czym jest kognitywistyka?, *Kognitywistyka i media w edukacji* 1/1998, Wydawnictwo Adam Marszałek, Toruń 1998.
- [47] Duch W., Korbicz J., Rutkowski L., Tadeusiewicz R., *Biocybernetyka i inżynieria biomedyczna*, tom 6: *Sieci neuronowe*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2000.

- [48] Duda R.O., Hart P.E., Stork D.G., *Pattern Classification*, John Wiley & Sons, Inc., Toronto 2001.
- [49] Elman J.L., Finding structure in time, *Cognitive Science*, **14**, 179–211 (1990).
- [50] Evolver — the Genetic Algorithm Problem Solver, Axcelis, Inc., 4668 Eastern Avenue N., Seattle, WA 98103, USA.
- [51] Fausett L., *Fundamentals of Neural Networks. Architectures, Algorithms, and Applications*, Prentice Hall, New Jersey 1994.
- [52] Ferber J., *Multi-Agent systems*, Addison-Wesley, New York 1999.
- [53] Findeisen W., Szymanowski W., Wierzbicki A., *Teoria i metody obliczeniowe optymalizacji*, PWN, Warszawa 1977.
- [54] FlexTool (GA) M2.1, Flexible Intelligence Group, L.L.C., Tuscaloosa, AL35486-1477, USA.
- [55] Fodor J.C., On fuzzy implication, *Fuzzy Sets and Systems*, **42**, 293–300 (1991).
- [56] Fogel D.B., *Evolutionary Computation. Towards a New Philosophy of Machine Intelligence*, IEEE Press, New York 1995.
- [57] Fraser A.S., Simulation of genetic systems by automatic digital computers. I. Introduction, Part I, II, *Aust. J. Biol. Sci.*, **10**, 484–499 (1957).
- [58] Fukami S., Mizumoto M., Tanaka K., Some considerations of fuzzy conditional inference, *Fuzzy Sets and Systems*, **4**, 243–273 (1980).
- [59] Galar R., *Miękka selekcja w losowej adaptacji globalnej w R^n . Próba biocybernetycznego ujęcia rozwoju*, Wydawnictwo Politechniki Wrocławskiej, Wrocław 1990.
- [60] Gen M., Cheng R., *Genetic Algorithms & Engineering Design*, John Wiley & Sons, Inc., New York 1997.
- [61] Giergel M.J., Hendzel Z., Żylski W., *Modelowanie i sterowanie mobilnych robotów kołowych*, Wydawnictwo Naukowe PWN, Warszawa 2002.
- [62] Główński C., Sztuczna inteligencja, *PC Kurier*, 14–27, 4 luty 1999.
- [63] Goldberg D.E., *Algorytmy genetyczne i ich zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa 1998.
- [64] Gorzałczany M.B., A interval-valued fuzzy inference method — some basic properties, *Fuzzy Sets and Systems*, **31**, 243–251 (1989).
- [65] Gorzałczany M.B., *Computational Intelligence Systems and Applications, Neuro-Fuzzy and Fuzzy Neural Synergisms*, Springer-Verlag, Heidelberg 2002.
- [66] Greco S., Matarazzo B., Słowiński R., Rough set processing of vague information using fuzzy similarity relations, *Finite Versus Infinite — Contributions to an Eternal Dilemma*, C.S. Calude i G. Paun (red.), Springer-Verlag, 149–173, London 2000.

- [67] Grzymała-Busse J.W., LERS — A system for learning from examples based on rough sets, w: Słowiński R. (red.), *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publishers, 3–18, Dordrecht 1992.
- [68] Hagan M., Menhaj M.B., Training feed forward networks with the Marquardt algorithm, *IEEE Trans. on Neural Networks*, **5**, 989–993 (1994).
- [69] Härdle W., *Applied Nonparametric Regression*, Cambridge University Press, London 1990.
- [70] Harel D., *Rzecz o istocie informatyki. Algorytmika*, wyd. 3, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- [71] Harris C.J., Moore C.G., Brown M., *Intelligent Control: Aspects of fuzzy logic and neural nets*, World Scientific, 1993.
- [72] Hassoun M.H., *Fundamentals of Artificial Neural Networks*, The MIT Press, Cambridge, Massachusetts 1995.
- [73] Haykin S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York 1994.
- [74] Hebb D., *Organization of behaviour*, John Wiley, New York 1949.
- [75] Helt P., Parol M., Piotrowski P., *Metody sztucznej inteligencji w elektroenergetyce*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2000.
- [76] Hertz J., Krogh A., Palmer R.G., *Wstęp do teorii obliczeń neuronowych*, Wydawnictwa Naukowo-Techniczne, Warszawa 1993.
- [77] Herrera F., Herrera E., Lozano M., Verdegay J.L., Fuzzy tools to improve genetic algorithms, w: *Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT'94)*, 1532–1539, Aachen 1994.
- [78] Herrera F., Lozano M., Verdegay J.L., Fuzzy connectives based crossover operators to model genetic algorithms population diversity, *Fuzzy Sets and Systems*, **92**(1), 21–30 (1997).
- [79] Hirota K. (red.), *Industrial Applications of Fuzzy Technology*, Springer-Verlag, Heidelberg, New York 1993.
- [80] Hisdal E., The IF THEN ELSE statement and interval-valued fuzzy sets of higher type, *Int. J. Man-Machine Studies*, 15, Academic Press Inc., 385–455, London 1981.
- [81] Hoffman P., Szach enter. Garri Kasparow kontra Deep Junior, *Wprost*, **57**, 9 luty 2003.
- [82] Holland J.H., *Adaptation in Natural and Artificial Systems*, Cambridge, The MIT Press, Cambridge, Massachusetts 1992.
- [83] Höppner F., Klawonn F., Kruse R., Runkler T., *Fuzzy cluster analysis. Methods for Classification, Data Analysis and Image Recognition*, John Wiley & Sons, Chichester 1999.

- [84] Hornik K., Approximation capabilities of multilayer feedforward networks, *Neural Networks*, **4**, 251–257 (1991).
- [85] Hornik K., Some new results on neural network approximation, *Neural Networks*, **6**, 1069–1072 (1993).
- [86] Hu Y.H., Hwang J. (red.), *Handbook of Neural Network Signal Processing*, CRC Press, Boca Raton 2002.
- [87] Inuiguchi M., Tanino T., *New Fuzzy Rough Sets Based on Certainty Qualification*, w: Pal S.K., Polkowski L., Skowron A. (red.), *Rough-Neural Computing: Techniques for Computing with Words*, Springer-Verlag, 277–296, Heidelberg, New York 2004.
- [88] Ishibuchi H., Nakashima T., Murata T., *40 Techniques and application of genetic algorithm-based methods for design compact fuzzy classification systems*, Fuzzy Theory Systems, Cornelius T. Leondes (red.), Academic Press, 1999.
- [89] Jagielski J., *Inżynieria wiedzy w systemach ekspertowych*, Lubelskie Towarzystwo Naukowe, Zielona Góra 2001.
- [90] Jang J.-S.R., ANFIS: Adaptive-network-based fuzzy inference systems, *IEEE Trans. Syst., Man, Cybern.*, **23**(3), 665–685 (1993).
- [91] Jankowski N., *Ontogeniczne sieci neuronowe. O sieciach zmieniających swoją strukturę*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2003.
- [92] Jordan M.I., Attractor dynamics and parallelism in a connectionist sequential machine, *Proc. of the Eight Annual Conference of the Cognitive Science Society*, 531–546, Hillsdale, NJ: Erlbaum 1986.
- [93] Kacprzak T., Ślot K., *Sieci neuronowe komórkowe*, Wydawnictwo Naukowe PWN, Warszawa 1994.
- [94] Kacprzyk J., *Zbiory rozmyte w analizie systemowej*, PWN, Warszawa 1986.
- [95] Kacprzyk J., *Wieloetapowe sterowanie rozmyte*, Wydawnictwa Naukowo-Tekniczne, Warszawa 2001.
- [96] Karayiannis N.B., Venetsanopoulos A.N., *Artificial Neural Networks*, Kluwer Academic Publishers, Boston, Dordrecht 1993.
- [97] Karnik N.N., Mendel J.M., *An Introduction to Type-2 Fuzzy Logic Systems*, University of Southern California, Los Angeles 1998.
- [98] Karnik N.N., Mendel J.M., Applications of type-2 fuzzy logic systems to forecasting of time-series, *Information Sciences* 120, Elsevier, 89–111, 1999.
- [99] Karnik N.N., Mendel J.M., Liang Q., Type-2 fuzzy logic systems, *IEEE Trans. on Fuzzy Systems*, **7**(6), 643–658 (1999).
- [100] Karnik N.N., Mendel J.M., Operations on type-2 fuzzy sets, *Fuzzy sets and systems*, **122**, 327–348 (2000).
- [101] Karnik N.N., Mendel J.M., Centroid of a type-2 fuzzy set, *Information Sciences* 132, Elsevier, 195–220, 2001.

- [102] Karr C.L., Design of an adaptive fuzzy logic controller using a genetic algorithm, *Proc. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA 1991.
- [103] Karr C.L., Genetic algorithms for fuzzy controllers, *AI Expert*, 1991, 26–33.
- [104] Kasabov N., *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, The MIT Press, Cambridge, Massachusetts 1996.
- [105] Kasperski M.J., *Sztuczna inteligencja*, Wydawnictwo Helion, Gliwice 2003.
- [106] Kawa K., Język sztucznej inteligencji, *PC Kurier*, 118–120, 18 czerwiec 1998.
- [107] Kay S.M., *Modern Spectral Estimation. Theory and Application*, Prentice Hall, Englewood Cliffs, New Jersey 1988.
- [108] Kecman V., *Learning and Soft Computing*, The MIT Press, Cambridge, Massachusetts 2001.
- [109] Khosla R., Dillon T., *Engineering intelligent hybrid multi-agent systems*, Kluwer Academic Publishers, Boston 1997.
- [110] Kitano H., Designing neural networks by genetic algorithms using graph generation systems, *Complex Systems*, 4, 461–476 (1990).
- [111] Klement E.P., Mesiar R., Pap E., *Triangular Norms*, Kluwer Academic Publishers, Netherlands 2000.
- [112] Köhn P., *Genetic Encoding Strategies for Neural Networks*, University of Tennessee — Universität Erlangen-Nürnberg, Dreibergstr. 5, 91056 Erlangen, Germany 1996.
- [113] Kohonen T., *Self-Organization and Associative Memory*, Springer-Verlag, Heidelberg, New York 1988.
- [114] Komosiński M., *Sztuczne życie*, Software 2.0, nr 2(86), 32–38, luty 2002.
- [115] Korbicz J., Obuchowicz A., Uciński D., *Sztuczne sieci neuronowe. Podstawy i zastosowania*, Akademicka Oficyna Wydawnicza 1994.
- [116] Korbicz J., Kościelny J.M., Kowalczyk Z., Cholewa W. (red.), *Diagnostyka procesów. Modele, metody sztucznej inteligencji, zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa 2002.
- [117] Kosiński R.A., *Sztuczne sieci neuronowe, dynamika nieliniowa i chaos*, Wydawnictwa Naukowo-Techniczne, Warszawa 2002.
- [118] Kosko B., *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ 1992.
- [119] Kościelny J.M., *Diagnostyka zautomatyzowanych procesów przemysłowych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001.
- [120] Koza J.R., *Genetic Programming II. Automatic Discovery of Reusable Programs*, The MIT Press, 1994.
- [121] Kröse B., Smagt P., *An introduction to Neural Networks*, Eight Ed., 1996.

- [122] Kruse R., Gebhardt J., Klawonn F., *Foundations of Fuzzy Systems*, John Wiley & Sons – Interscience, New York 1994.
- [123] Krawiec K., Stefanowski J., *Uczenie maszynowe i sieci neuronowe*, Wydawnictwo Politechniki Poznańskiej, Poznań 2003.
- [124] Kurowski J., Człowiek kontra komputer — Deep Blue — prawdy, mity, co dalej, *Software* 2.0, nr 2(86), 26–31, luty 2002.
- [125] Lee M.A. Takagi H., *Dynamic control of genetic algorithms using fuzzy logic techniques*, Fifth International Conference on Genetic Algorithms (ICGA' 93), 76–83, Urbana-Champaign 1993.
- [126] Li H., Chen C.L.P., Huang H., *Fuzzy Neural Intelligent Systems. Mathematical Foundation and the Applications in Engineering*, CRC Press, Boca Raton 2001.
- [127] Liu J., *Autonomous agents and multi-agent systems*, World Scientific, Singapore 2001.
- [128] Lowen R., *Fuzzy Set Theory Basic Concepts, Techniques and Bibliography*, Kluwer Academic Publishers, Dordrecht 1996.
- [129] Luger G.F., *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Fifth Ed., Addison-Wesley, London 2005.
- [130] Lachwa A., *Rozmyty świat zbiorów, liczb, relacji, faktów, reguł i decyzji*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001.
- [131] Mańdziuk J., *Sieci neuronowe typu Hopfielda. Teoria i przykłady zastosowań*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2000.
- [132] Marple S.L. Jr., *Digital Spectral Analysis with applications*, Prentice Hall, Englewood Cliffs, New Jersey 1987.
- [133] McCulloch W.S., A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5, 115–133 (1943).
- [134] Mendel J.M., *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice Hall PTR, London 2001.
- [135] Menzel P., D'Aluisio F., *Robo Sapiens*, Wydawnictwo Gruner+Jahr Polska, Warszawa 2002.
- [136] Michalewicz Z., *Algorytmy genetyczne + Struktury danych = Programy ewolucyjne*, Wydawnictwa Naukowo-Techniczne, Warszawa 1999.
- [137] Miller G.F., Todd P.M., Hagde S.U., Designing neural networks using genetic algorithms, *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, Schaffer J.D. (red), Morgan Kauffman, 379–384, San Mateo, CA 1989.
- [138] Minsky M., Papert S., *Perceptrons: an introduction to computational geometry*, Cambridge, MA, 1988.

- [139] Montana D., Davis L., Training feedforward neural networks using genetic algorithms, *Proceedings of 11th International Conference on Artificial Intelligence*, 762–767, Morgan Kauffman, 1989.
- [140] Mrózek A., Płonka L., *Analiza danych metodą zbiorów przybliżonych. Zastosowania w ekonomii, medycynie i sterowaniu*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.
- [141] Mulawka J., *Systemy ekspertowe*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
- [142] Nauck D., Klawon F., Kruse R., *Foundations of Neuro-Fuzzy Systems*, John Wiley & Sons – Interscience, New York 1997.
- [143] Nęcka E., *Inteligencja. Geneza, struktura, funkcje*, Gdańskie Wydawnictwo Psychologiczne, Gdańsk 2003.
- [144] Nguyen H.T., Walker E.A., *A First Course in Fuzzy Logic*, Second Ed., Chapman & Hall/CRC, Boca Raton 2000.
- [145] Nie J., Linkens D., *Fuzzy-Neural Control*, Prentice Hall, Englewood Cliffs, NJ 1995.
- [146] Niederliński A., *Regułowe systemy ekspertowe*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 2000.
- [147] Nilsson N.J., *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, San Francisco 1998.
- [148] Nowicki R., *Systemy rozmyto-neuronowe realizujące różne sposoby rozmytego wnioskowania*, Praca doktorska, AGH, Kraków 1999.
- [149] Nowicki R., Rutkowski L., Rough-Neuro-Fuzzy System for Classification, *Proc. of Fuzzy Systems and Knowledge Discovery*, 149, Singapore 2002.
- [150] Nowicki R., Rutkowski L., *Przybliżone zbiorы rozmyte w diagnostyce*, VI Krajowa Konferencja Naukowo-Techniczna Diagnostyka Procesów Przemysłowych, PWNT, 67–72, Gdańsk 2003.
- [151] Nowicki R., *Rough Sets in the Neuro-Fuzzy Architectures Based on Monotonic Fuzzy Implications*, Lecture Notes in Artificial Intelligence, nr 3070, Artificial Intelligence and Soft Computing, Rutkowski L., Siekmann J., Tadeusiewicz R., Zadeh L.A. (red.), Springer-Verlag, 510–517, Berlin, Heidelberg 2004.
- [152] Nowicki R., *Rough Sets in the Neuro-Fuzzy Architectures Based on Non-monotonic Fuzzy Implications*, Lecture Notes in Artificial Intelligence, nr 3070, Artificial Intelligence and Soft Computing, Rutkowski L., Siekmann J., Tadeusiewicz R., Zadeh L.A. (red.), Springer-Verlag, 518–525, Berlin, Heidelberg 2004.
- [153] Obuchowicz A., *Evolutionary Algorithms for Global Optimization and Dynamic System Diagnosis*, Lubusky Scientific Society in Zielona Góra, Zielona Góra 2003.

- [154] Osowski Ł., Budujemy własny syntezator mowy, *Software 2.0*, nr 2(98), 26–35, luty 2003.
- [155] Osowski S., *Sieci neuronowe w ujęciu algorytmicznym*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
- [156] Osowski S., *Sieci neuronowe do przetwarzania informacji*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2000.
- [157] Pagan A., Ullah A., *Nonparametric Econometrics*, Cambridge University Press, London 1999.
- [158] Pal S.K., Polkowski L., Skowron A., *Rough-Neural Computing, Techniques for Computing with Words*, Springer-Verlag, Berlin 2004.
- [159] Parzen E., On estimation of probability density function and mode, *Annals Mathematics of Statistics*, **33**, 1065–1076 (1962).
- [160] Pawlak M., *Algorytmy ewolucyjne jako narzędzie harmonogramowania produkcji*, Wydawnictwo Naukowe PWN, Warszawa 1999.
- [161] Pawlak Z., Rough sets, *Proc. on Int. Journal of Information and Computer Science*, **11**, 341 (1982).
- [162] Pawlak Z., *Systemy informacyjne. Podstawy teoretyczne*, Wydawnictwa Naukowo-Techniczne, Warszawa 1983.
- [163] Pawlak Z., *Rough Sets — Theoretical Aspect of Reasoning About Data*, Kluwer Academic Publishers, Dordrecht 1991.
- [164] Pawlak Z., Rough sets, decision algorithms and Bayes' theorem, *European Journal of Operational Research*, **136**, 181–189 (2002).
- [165] Pedrycz W., *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons – Interscience, New York 1993.
- [166] Pedrycz W., *Fuzzy Sets Engineering*, CRC Press, Inc., Boca Raton 1995.
- [167] Pedrycz W., Gomide F., *An Introduction to Fuzzy Sets: Analysis and Design*, A Bradford Book, The MIT Press, Cambridge, Massachusetts 1998.
- [168] Penrose R., *Nowy umysł cesarza. O komputerach, umyśle i prawach fizyki*, Wydawnictwo Naukowe PWN, Warszawa 1995.
- [169] Perta R., Inwazja mutantów, *Chip*, **9**, 106–110 (2003).
- [170] Pieczyński A., *Reprezentacja wiedzy w diagnostycznym systemie ekspertowym*, Lubelskie Towarzystwo Naukowe w Zielonej Górze, Zielona Góra 2003.
- [171] Piegat A., *Modelowanie i sterowanie rozmyte*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 1999.
- [172] Piliński M., *Program FLiNN — Podręcznik użytkownika*, Polskie Towarzystwo Sieci Neuronowych, Częstochowa 1996.
- [173] Piliński M., *Sterowniki rozmyte z wykorzystaniem sieci neuronowych*, Praca doktorska, Politechnika Wrocławskiego, 1996.

- [174] Piliński M., Universal network trainer, *Proc. of the Second Conference, Neural Networks and Their Applications*, 2, 383–391, Częstochowa 1996.
- [175] Podsiadło M., Uwagi do twierdzenia o schematach, *Materiały I Krajowej Konferencji: Algorytmy Ewolucyjne*, Politechnika Warszawska, 1996.
- [176] Pokropińska A., *Badanie efektywności działania systemów neuronowo-rozmytych*, Praca doktorska, Politechnika Częstochowska, Częstochowa 2005.
- [177] Polkowski L., *Rough Sets, Mathematical Foundation*, Springer-Verlag, Heidelberg, New York 2002.
- [178] Principe J.C., Euliano N.R., Lefebvre W.C., *Neural and Adaptive Systems*, John Wiley & Sons, New York 2000.
- [179] Radosiński E., *Systemy informatyczne w dynamicznej analizie decyzyjnej*, Wydawnictwo Naukowe PWN, Warszawa–Wrocław 2001.
- [180] Radzikowska A.M., Kerre E.E., A comparative study of fuzzy rough sets, *Fuzzy sets and systems*, 126, 137–155 (2002).
- [181] Rosenblatt F., *On the Convergence of Reinforcement Procedures In Simple Perceptrons*, Cornell Aeronautical Laboratory Report VG-1196-G-4, Buffalo, NY 1960.
- [182] Rudeński A., Zastosowanie metod deterministycznych oraz ewolucyjnych w zadanach optymalizacji szczegółowej i ogólnej maszyn elektrycznych. *Proc. of XL International Symposium on Electrical Machines SME-2004*, 200–204, 2004.
- [183] Rumelhart D.E., Hinton G.E., Williams R.J., Learning internal representations by error propagation, w: *Parallel Distributed Processing*, vol. 1, ch. 8, Rumelhart D.E. and McClelland J.L. (red.), The MIT Press, Cambridge, Massachusetts 1986.
- [184] Russell S.J., Norvig P., *Artificial Intelligence: A Modern Approach (2nd Ed.)*, Prentice Hall, 2002.
- [185] Rutkowska D., *Inteligentne systemy obliczeniowe. Algorytmy genetyczne i sieci neuronowe w systemach rozmytych*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1997.
- [186] Rutkowska D., Piliński M., Rutkowski L., *Sieci neuronowe algorytmy genetyczne i systemy rozmyte*, Wydawnictwo Naukowe PWN, Warszawa 1999.
- [187] Rutkowska D., *Neuro-Fuzzy Architectures and Hybrid Learning*, Physica-Verlag, Springer-Verlag Company, Heidelberg, New York 2002.
- [188] Rutkowski L., Sequential estimates of probability densities by orthogonal series and their application in pattern classification, *IEEE Trans. on Systems, Man, and Cybernetics*, 10(12), 918–920 (1980).
- [189] Rutkowski L., Sequential estimates of a regression function by orthogonal series with applications in discrimination, w: *Lectures Notes in Statistics*, Springer-Verlag, vol. 8, 236–244, Heidelberg, New York 1981.

- [190] Rutkowski L., On Bayes risk consistent pattern recognition procedures in a quasi-stationary environment, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **PAMI-4**(1), 84–87 (1982).
- [191] Rutkowski L., On system identification by nonparametric function fitting, *IEEE Trans. on Automatic Control*, **AC-27**, 225–227 (1982).
- [192] Rutkowski L., On-line identification of time-varying systems by nonparametric techniques, *IEEE Trans. on Automatic Control*, **AC-27**, 228–230 (1982).
- [193] Rutkowski L., On nonparametric identification with prediction of time-varying systems, *IEEE Trans. on Automatic Control*, **AC-29**, 58–60 (1984).
- [194] Rutkowski L., Nonparametric identification of quasi-stationary systems, *Systems and Control Letters*, **6**, 33–35 (1985).
- [195] Rutkowski L., The real-time identification of time-varying systems by nonparametric algorithms based on the Parzen kernels, *International Journal of Systems Science*, **16**, 1123–1130 (1985).
- [196] Rutkowski L., Sequential pattern recognition procedures derived from multiple Fourier series, *Pattern Recognition Letters*, **8**, 213–216 (1988).
- [197] Rutkowski L., Rafajłowicz E., On global rate of convergence of some nonparametric identification procedures, *IEEE Trans. on Automatic Control*, **AC-34**, nr 10, 1089–1091 (1989).
- [198] Rutkowski L., Nonparametric learning algorithms in the time-varying environments, *Signal Processing*, **18**, 129–137 (1989).
- [199] Rutkowski L., An application of multiple Fourier series to identification of multivariable nonstationary systems, *Int. Journal of Systems Science*, **20**(10), 1993–2002 (1989).
- [200] Rutkowski L., Identification of MISO nonlinear regressions in the presence of a wide class of disturbances, *IEEE Trans. on Information Theory*, **IT-37**, 214–216 (1991).
- [201] Rutkowski L., Multiple Fourier series procedures for extraction of nonlinear regressions from noisy data, *IEEE Trans. on Signal Processing*, **41**(10), 3062–3065 (1993).
- [202] Rutkowski L., *Filtры адаптационные и адаптационное преобразование сигналов: теория и applications*, Wydawnictwa Naukowo-Techniczne, Warszawa 1994.
- [203] Rutkowski L., Gałkowski T., On pattern classification and system identification by probabilistic neural networks, *Appl. Math. and Comp. Sci.*, **4**(3), 413–422 (1994).
- [204] Rutkowski L., *Sieci neuronowe i neurokomputery*, Wydawnictwo Politechniki Częstochowskiej, Częstochowa 1996.
- [205] Rutkowski L., Cierniak R., Image compression by competitive learning neural networks and predictive vector quantization, *Appl. Math. and Comp. Science*, **6**(3), 431–445 (1996).

- [206] Rutkowski L., Tadeusiewicz R. (red.), *Proc. of the Fourth Conference: Neural Networks and Their Applications*, Zakopane 1999.
- [207] Rutkowski L., Cierniak R., On image compression by competitive neural networks and optimal linear predictors, *Signal Processing: Image Communication — a Eurosim Journal*, Elsevier Science B.V., February, vol. 15, nr 6, 559–565, Amsterdam 2000.
- [208] Rutkowski L., Cpałka K., Flexible structures of neuro-fuzzy systems, *Quo Vadis Computational Intelligence, Studies in Fuzziness and Soft Computing*, Springer-Verlag, vol. 54, 479–484, Heidelberg, New York 2000.
- [209] Rutkowski L., Tadeusiewicz R. (red.), *Proc. of the Fifth Conference: Neural Networks and Soft Computing*, Zakopane 2000.
- [210] Rutkowski L., Cpałka K., A neuro-fuzzy controller with a compromise fuzzy reasoning, *Control and Cybernetics*, **31** (2), 297–308, 2002.
- [211] Rutkowski L., Cpałka K., Compromise approach to neuro-fuzzy systems, *Proc. of the 2nd Euro-International Symposium on Computational Intelligence*, vol. 76, 85–90, Koszyce 2002.
- [212] Rutkowski L., Cpałka K., Elastyczne systemy rozmyto-neuronowe, *XIV Krajowa Konferencja Automatyki*, 813–818, 24–27 czerwca, Zielona Góra 2002.
- [213] Rutkowski L., Scherer R., Nowe struktury rozmyto-neuronowe, *XIV Krajowa Konferencja Automatyki*, 809–812, 24–27 czerwca, Zielona Góra 2002.
- [214] Rutkowski L., Starczewski J., Nowe metody modelowania niepewności w systemach rozmytych, *XIV Krajowa Konferencja Automatyki*, 885–888, 24–27 czerwca, Zielona Góra 2002.
- [215] Rutkowski L., Cpałka K., Flexible weighted neuro-fuzzy systems, w: *Proc. 9th Int. Conf. on Neural Information Processing (ICONIP'02)*, Orchid Country Club, November 18–22, Singapore 2002.
- [216] Rutkowski L., Nowicki R., Hybrid soft computing techniques and their applications, *Proc. of the Symposium on Methods of Artificial Intelligence AI-METH*, 55–60, November 13–15, Gliwice 2002.
- [217] Rutkowski L., Cpałka K., Compromise weighted neuro-fuzzy systems, w: Rutkowski L., Kacprzyk J. (red.), *Neural Networks and Soft Computing*, Heidelberg, Springer-Verlag, 557–562, Heidelberg, New York 2003.
- [218] Rutkowski L., Cpałka K., Flexible neuro-fuzzy systems, *IEEE Trans. on Neural Networks*, **14**, 554–574 (2003).
- [219] Rutkowski L., Kacprzyk J. (red.), *Neural Networks and Soft Computing*, Springer-Verlag, Heidelberg, New York 2003.
- [220] Rutkowski L., Cpałka K., Neuro-fuzzy systems derived from quasi-triangular norms, *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, 1031–1036, July 26–29, Budapest 2004.

- [221] Rutkowski L., Adaptive probabilistic neural-networks for pattern classification in time-varying environment, *IEEE Trans. Neural Networks*, **15**, 811–827 (2004).
- [222] Rutkowski L., Generalized regression neural networks in time-varying environment, *IEEE Trans. Neural Networks*, **15**, 576–596 (2004).
- [223] Rutkowski L., A New Method for System Modelling and Pattern Classification, *Bulletin of the Polish Academy of Sciences*, **52**(1), 11–24 (2004).
- [224] Rutkowski L., New Soft Computing Techniques for System Modelling, w: *Pattern Classification and Image Processing*, Springer-Verlag, Heidelberg and New York, 2004.
- [225] Rutkowski L., *Flexible Neuro-Fuzzy Systems: Structures, Learning and Performance Evaluation*, Kluwer Academic Publishers, Boston, Dordrecht 2004.
- [226] Rutkowski L., Siekmann J., Tadeusiewicz R., Zadeh L. A., (red.): *Proc. of the Seventh Int. Conf. Artificial Intelligence and Soft Computing*, ICAISC-2004, Zakopane, Springer-Verlag, Heidelberg, New York 2004.
- [227] Rutkowski L., Cpałka K., Designing and learning of adjustable quasi-triangular norms with applications to neuro-fuzzy systems, *IEEE Trans. on Fuzzy Systems*, **14**, 140–151 (2005).
- [228] Rzewuski M., Zabawa w stwórcę, *PC Kurier*, **19**, 28–37 (2002).
- [229] Sadowski W., Głowa atomowa, *Polityka*, Nr 16 (2397), 19 kwiecień 2003.
- [230] Schaffer J.D., Whitley L., Eshelman J., Combinations of genetic algorithms and neural networks, a survey of the state of the art, *Proc. of Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, COGANN-92, 1992.
- [231] Scherer R., *Methods of Classification Using Neuro-Fuzzy Systems*, Praca doktorska, Politechnika Częstochowska, Częstochowa 2002.
- [232] Shonkwiler R., Miller K.R., Genetic algorithm, neural network synergy for non-linearly constrained optimization problems, *Proc. of Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, COGANN-92, 248–257, 1992.
- [233] Słowiński R. (red.), *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publishers, Boston, Dordrecht 1992.
- [234] Słupecki J., Hałkowska K., Piróg-Rzepecka K., *Logika i teoria mnogości*, Wydawnictwo Naukowe PWN, Warszawa 1994.
- [235] Söderström T., Stoica P., *Identyfikacja systemów*, Wydawnictwo Naukowe PWN, Warszawa 1997.
- [236] Specht D.F., Probabilistic neural networks, *Neural Networks*, **3**, 109–118 (1990).
- [237] Specht D.F., A general regression neural network, *IEEE Trans. on Neural Networks*, **2**, 568–576 (1991).

- [238] Starczewski J., *Rozmyte systemy wnioskujące typu 2*, Praca doktorska, Politechnika Częstochowska, Częstochowa 2002.
- [239] Starczewski J., Rutkowski L., Neuro-Fuzzy Systems of Type 2, *1st Int'l Conf. on Fuzzy Systems and Knowledge Discovery*, vol. 2, 458–462, Singapore 2002.
- [240] Starczewski J., What differs interval type-2 FLS from type-1 FLS, w: Rutkowski L. (red.) *Int'l Conf. on Artificial Intelligence & Soft Computing*, Zakopane, June 2004, *Lecture Notes in Computer Science*, Springer, 2004.
- [241] Tadeusiewicz R., *Problemy biocybernetyki*, Wydawnictwo Naukowe PWN, Warszawa 1991.
- [242] Tadeusiewicz R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993.
- [243] Tadeusiewicz R., Rutkowski L., Chojcan J. (red.), *Proc. of the Third Conference: Neural Networks and Their Applications*, Kule 1997.
- [244] Tadeusiewicz R., *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.
- [245] Tadeusiewicz R., Ogiela M.R., *Medical Image Understanding Technology*, Springer-Verlag, Heidelberg 2004.
- [246] Takagi T., Sugeno M., Fuzzy identification of systems and its application to modeling and control, *IEEE Trans. Systems, Man, and Cybernetics*, **15**, 116–132 (1985).
- [247] Takayama H., Intruzi w pałacu, *Newsweek*, 92, 12 styczeń 2003.
- [248] Tettamanzi A., Tomassini M., *Soft Computing. Integrating Evolutionary, Neural, and Fuzzy Systems*, Springer-Verlag, Berlin, Heidelberg 2001.
- [249] Valenzuela-Rendon M., The fuzzy classifier system: A classifier system for continuously varying variables, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 346–353, Morgan Kaufmann, San Mateo 1991.
- [250] Volna E., Genetic Search of Optimal Neural Network Topology for problem of Pattern Recognition, *Fifth Conference Neural Networks and Soft Computing*, 657–662, Zakopane 2000.
- [251] Volna E., Learning Algorithm of Feedforward Neural Networks, *Fourth Conference Neural Networks and Their Applications*, 611–616, Zakopane 1999.
- [252] Wang L.-X., Mendel J.M., Generating fuzzy Rules by Learning from Examples, *IEEE Trans. Systems, Man, and Cybernetics*, **22**(6), 1414–1427, 1992.
- [253] Wang L.-X., *Adaptive Fuzzy Systems and Control, Design and Stability Analysis*, Prentice Hall PTR, Upper Saddle River, NJ 1994.
- [254] Wang L., Yen J., Application of statistical information criteria for optimal fuzzy model construction, *IEEE Trans. on Fuzzy Systems*, **6**, 353–362 (1998).

- [255] Weiss G. (red.), *Multiagent systems. A modern approach to distributed artificial intelligence*, The MIT Press, Cambridge 1999.
- [256] Whitley D., Applying genetic algorithms to neural network learning, *Proceedings of the Seventh Conference of the Society of Artificial Intelligence and Simulation of Behavior*, Sussex, England, Pitman Publishing, 137–144, 1989.
- [257] Widrow B., Hoff M.E. Jr., Adaptive switching circuits, *Western Conf. Rec.*, IRE, Part 4, 94–104, 1960.
- [258] Widrow B., Stearns S., *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ 1985.
- [259] Widrow B., Lehr M.A., 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation, *Proc. of the IEEE*, **78**(9), 1415–1442 (1990).
- [260] Winter G., Périaux J., Galán M., Cuesta P., *Genetic algorithms in Engineering and Computer Science*, John Wiley & Sons, Chichester 1995.
- [261] Wu H., Mendel J.M., Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems, *IEEE Trans. Fuzzy Syst.*, **10**(5), 622–639 (2002).
- [262] Yager R.R., Filev D.P., *Podstawy modelowania i sterowania rozmytego*, Wydawnictwa Naukowo-Techniczne, Warszawa 1995.
- [263] Yao X., A review of evolutionary artificial neural networks, *Int. Journal of Intelligent Systems*, **1993**, 539–567.
- [264] Yao X., Liu Y., Towards designing artificial neural networks by evolution, *Applied Mathematics and Computation*, **91**(1), 83–90 (1998).
- [265] Zadeh L.A., Fuzzy sets, *Information and Control*, **8**, 338–353 (1965).
- [266] Zadeh L.A., The concept of a linguistic variable and its application to approximate reasoning, Parts 1–3, *Inf. Sci.*, no. 8, 199–249; no. 8, 301–357; no. 9, 43–80; 1975.
- [267] Zalewski A., Cegiełka R., *Matlab — obliczenia numeryczne i ich zastosowania*, Wydawnictwo NAKOM, Poznań 2002.
- [268] Zieliński J.S. (red.), *Inteligentne systemy w zarządzaniu. Teoria i praktyka*, Wydawnictwo Naukowe PWN, Warszawa 2000.
- [269] Zimmermann H.-J., *Fuzzy Set Theory*, Kluwer Academic Publishers, Boston, Dordrecht 1994.
- [270] Żurada J., *Introduction to Artificial Neural Systems*, West Publishing Company, 1994.
- [271] Żurada J., Barski M., Jędruch W., *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, Warszawa 1996.

SKOROWIDZ

- Adaptacyjne algorytmy ewolucyjne** 288
 - algorytm FCM 308
 - FMLE 313
 - gazu neuronowego 212
 - genetyczny 232
 - Gustafsona–Kessela 311
 - HCM 307
 - Levenberga–Marquardta 189
 - momentum 187
 - mrówkowy 15
 - OBD 198
 - OBS 199
 - PCM 310
 - RLS 191
 - wstępnej propagacji błędów 185
 - WTA 207
 - WTM 212
 - zmiennej metryki 188
 - allel 231, 236
 - aproksymacja rodziny zbiorów dolna 36
 - – – góra 37
 - – –, jakość 38
 - zbioru dolna 28, 29
 - – góra 29, 30
 - architektura sieci 193
 - atrybut 20, 21
 - decyzyjny 24, 39, 46
 - nieusuwalny 43
 - niezależny 42
 - warunkowy 24, 39, 46
 - zależny 42
 - zbędny 43
 - Biomorfy** 16
 - błąd średni kwadratowy 167
 - bot 17, 18
 - Alice 18
 - Eliza 18
 - Cecha** 20, 21, 24, 301
 - cel rozmyty 121, 122
 - chromosom 231, 236
 - Decyzja rozmyta** 121, 122
 - dokładność aproksymacji 35, 36, 38
 - Efektywność działania systemu** 373
 - eksploracja 269, 288
 - eksploracja 269, 288
 - ewolucja systemów rozmytych 290
 - Fenotyp** 231
 - funkcja celu 231
 - informacyjna 21, 23
 - przynależności 52
 - dolnej 137
 - – drugorzędnej 133, 135
 - – głównej 135
 - – górnej 137
 - przystosowania 231
 - testowa Ackleya 259
 - – Rastingira 265
 - Gen** 231
 - generacja 232
 - generator archimedesowej *t*-normy addytywnej 85
 - – – multiplikatywny 85
 - genotyp 231, 236
 - Grzymała–Busse Jerzy 51
 - Heterogeniczność** 300
 - heurystyka 12
 - hipoteza cegiełek 250
 - sztucznej inteligencji mocna 9
 - – – słaba 9
 - homogeniczność 300

- Iloczyn kartezjański zbiorów rozmytych 67, 89, 144
 iloraz zbioru przez relację 25
 implikacja rozmyta 100
 inwersja 279
 Jakość aproksymacji rodziny zbiorów 38
Klasa abstrakcji 25–27
 kognitywistyka 13, 14
 kryterium Akaike (AIC) 374
 – CAT 375
 – Fukuyamy–Sugeno 315
 – jakości grupowania 314
 – końcowego błędu predykcji (FPE) 375
 – rozmycia macierzy podziału 314
 – Schwarza 375
 – Söderströma i Stoicy 375
 – Xie–Bieni 315
 krzyżowanie 234, 255, 277
 – dwupunktowe 277
 – jednopunktowe 234
 – równomierne 278
 – wielopunktowe 277
 kwantyzacja danych 49
- Linie izokryterialne 373
 locus 231
- Łańcuchy 231, 236
- Metoda Delphi 114
 – – rozmyta 114
 – – ważona rozmyta 117, 118
 – kodowania 274
 – – binarna 274
 – – logarytmiczna 275
 – – zmiennoprzecinkowa 275
 – PERT 119
 – – rozmyta 118
 – rozpadu wag 198
 – selekcji 232, 269
 – –, koło ruletki 233, 237, 240, 270
 – – progowa 272
 – – rankingowa 270
 – – stłoczenia 272
 – – turniejowa 271
 – ścieżki krytycznej 118
 miara diagonalna 306
 – Euklidesowa 300, 305, 306
 – Hamminga 305
 – Mahalanobisa 307
 – Manhattan 305
 – Minkowskiego 305, 306
 model lingwistyczny 104
- modelowanie nieliniowego obiektu dynamicznego (NDP) 320
 – smaku ryżu 320
 – statycznej funkcji nieliniowej (HANG) 320
 mutacja 234, 235, 256, 279
 – binarna 279
 – liczb zmiennoprzecinkowych 279
- Nacisk selektywny 269, 288
 negacja 89
 neuron Adaline 167
 – Hebb 177
 – perceptron 161
 – sigmoidalny 172
 normy trójkątne dualne 82
 nośnik zbioru rozmytego 60
- Obiekt** 20
 – fizyczny 20
 obszar brzegowy 31–33, 37
 – negatywny 32, 33, 38
 – pozytywny 31, 37
 – rozważań 52
 ograniczenie rozmyte 121, 122
 otoczenie rozmyte 121, 122
- Pawlak Zdzisław 50
 podejście Michigan 290, 294
 – Pittsburgh 290, 295
 podział ostry 302
 – posybilistyczny 304
 – rozmyty 303
 pokolenie 232
 polimeryzacja 319
 populacja 231
 – rodzicielska 234
 problem plecakowy 242
 proces autoregresji 373
 program ASTER 11
 – DENDRAL 7
 – LERS 45–50
 programowanie ewolucyjne 266
 – genetyczne 266
 przecięcie zbiorów rozmytych typu 1 142
 – – typu 2 141, 142
 przestrzeń rozważań 20
 przystosowanie schematu 245
 pula rodzicielska 234, 238
 punkt krzyżowania 234
- Rdzeń** zbioru atrybutów 43
 redukcja typu 146, 147, 151
 redukt zbioru atrybutów 43, 45
 – względny 43
 reguła 24
 – Hebb 177

- reguła największego spadku 182
 - 1/5 sukcesów 251
 - wnioskowania *modus ponens* 94
 - – – uogólniona 95
 - – – *tollens* 95
 - – – uogólniona 98
- reguły deterministyczne 39
 - –, generowanie 46
 - –, naprawa 39
 - niedeterministyczne 39
- relacja nieroróżnialności 25
 - rozmyta 90, 144
 - – binarna 144
 - równoważności 25
- robot AIBO 9
 - ASIMO 9
 - Cog 10
 - Kismet 10
- rodzina zbiorów 25
- rozmiar systemu 373
- rozmywanie typu 1 151–153
 - typu 2 151–153
- rozpiętość schematu 247
- rząd schematu 246
- Samoadaptacja 251, 254, 256
- schemat 244
 - sieć neuronowa 280
 - –, projektowanie 283, 285
 - –, uczenie 280, 285
 - sieci ART 216
 - BAM 205
 - Elmana 205
 - Hamminga 203
 - Hopfielda 201
 - Kohonena 213
 - probabilistyczne 225
 - rekurencyjne 199
 - RTRN 205
 - samoorganizujące 206
 - singleton 55
- skalowanie funkcji przystosowania liniowe 272
 - – –, obcinanie typu sigma 273
 - – – potęgą 273
- stopień aktywacji reguły 154–156
 - drugorzędnej przynależności 133, 135
 - zależności zbioru atrybutów 39
- strategia elitarna 273
- strategie ewolucyjne 250
 - – (1 + 1) 251
 - – ($\mu + \lambda$) 254
 - – (μ, λ) 261
- suma zbiorów rozmytych 63, 88, 140–142
- synteza mowy 10
- system Framstick 16
- informacyjny 21–22
- Luna 11
- MYCIN 8
- NEOMYCIN 8
- PROSPEKTOR 7
- systemy ekspertowe 7, 8
 - neuronowo-rozmyte 318
 - – typu A 322
 - – – B 324
 - – – logicznego 335
 - – – Mamdaniego 321
 - – – Takagi–Sugeno 352
 - wnioskujące rozmyte 151, 152
- Ślad niepewności 136
- Tablica decyzyjna 24–25
 - – dobrze określona 39, 41
 - –, naprawa 39
 - – źle określona 39
 - t*-konorma 81
 - ważona 81
 - t*-norma 80
 - ważona 81
 - tłumaczenie maszynowe 11
 - twierdzenie Kołmogorowa 193
 - o dekompozycji 64
 - o schematach 244, 249
- Uaktualnianie wag kumulacyjne 185
 - – przyrostowe 185
- uczenie iteracyjne 290, 297
 - nadzorowane 163
- Wagi 160
 - wartość atrybutu 20
 - cechy 20
 - współczynnik istotności 44
 - wymiana populacji częściowa 273
 - wymiar Vapnika–Chervonenkisa 197
- Zasada oszczędności 373
 - zbiór definiowalny 34
 - dokładny 34
 - niedefiniowalny całkowicie 35
 - – wewnętrznie 34
 - – zewnętrznie 35
 - rozmyty 121
 - – osadzony typu 1 138–140, 142
 - – – typu 2 138–142
 - –, przecięcie 63, 88
 - – wklęsły 63
 - – wypukły 62
 - złożenie rozszerzone 146