Politechnika Śląska
Wydział Autmatyki, Elektroniki i Informatyki
Instytut Informatyki

AUTOREFERAT ROZPRAWY DOKTORSKIEJ

Marcin Blachnik

Systemy regułowe bazujące na prototypach oraz ich relacje z systemami rozmytymi w zastosowaniu do klasyfikacji danych

Promotor:
dr. hab. Tadeusza Wieczorka
Prof. Politechniki Śląskiej

Gliwice, 2007

Podziękowania

Pragnę wyrazić swoją wdzięczność promotorowi mojej pracy Panu Prof. Tadeuszowi Wieczorkowi, a także Kolegom z Katedry, w szczególności Jackowi Biesiadzie za wiele inspirujących dyskusji i konstruktywnej krytyki. Bardzo duże wyrazy uznania należą się również Panu Prof. Włodzisławowi Duchowi oraz jego Współpracownikom za bezinteresowną pomoc i merytoryczne wsparcie podczas realizacji badań.
Z całego serca dziękuję również moim najbliższym i przyjaciołom, którzy mnie nieustannie wzmacniali, w szczególności mojej żonie Ani, która dzielnie znosiła wszystkie ciężary, jakie na nią spadły zwłaszcza w końcowym etapie moich prac nad rozprawą. Na koniec pragnę podziękować moim rodzicom, którzy zainteresowali mnie nauką oraz zaszczepili ciekawość badawczą.

Spis treści

Rozdział 1

Wstęp

W obecnych czasach ilość gromadzonych danych w każdej niemal dziedzinie jest ogromna, jednakże nie jest ona skorelowana z wiedzą, którą posiadamy na temat badanych zjawisk. Co więcej, często problemem staje się nadmierna liczba danych, których powszechnie używane systemy nie potrafią przeanalizować, nie mówiąc już o możliwości wydobycia ukrytej w nich wiedzy.
Problem stanowi również analiza wiedzy ekspertów, którzy na podstawie intuicji podejmują pewne działania, nie potrafiąc racjonalnie wytłumaczyć przesłanek wiodących do określonej konkluzji. Dlatego też istotna jest budowa systemów pozwalających na wydobywanie zgromadzonej w danych wiedzy w postaci zbiorów reguł. Problem ten przez społeczność naukową analizowany jest różnymi metodami określanymi ogólnie mianem systemów *inteligencji obliczeniowej* (ang. computational intelligence, CI). Systemy te grupują różne podejścia do problemu analizy danych typu metody statystyczne, probabilistyczne, uczenie maszynowe, systemy rozmyte, metody zbiorów przybliżonych, sztuczne sieci neuronowe oraz wnioskowanie w oparciu o przypadki (ang. case based reasonong).
Należy jednak zwrócić uwagę na zaproponowane w pracach W. Ducha i współpracowników systemy bazujące na podobieństwie (ang. similarity based methods, SBM) [49, 45]. W pracach tych autorzy pokazali możliwości jakie daje uczenie w oparciu o podobieństwo jako narzędzie do generalizacji i integracji różnych systemów inteligencji obliczeniowej. Między innymi w pracach [48, 45, 44] zaprezentowana została możliwość integracji sieci neuronowych typu perceptron wielowarstwowy (MLP) z systemami SBM.
Bardzo istotnym zagadnieniem pozostaje problem reprezentacji wiedzy w postaci zrozumiałej dla człowieka. Wiele z istniejących systemów inteligencji obliczeniowej pozbawione jest tej możliwości, co w pewnych zagadnieniach ogranicza ich użyteczność. Przykładem tego mogą być zastosowania medyczne, gdzie proces wnioskowania przez system - proces diagnozowania schorzeń, musi posiadać właściwość interpretacji podjętej decyzji. Podobna sytuacja występuje w przypadku pewnych zagadnień automatycznego sterowania urządzeniami, jak sterowaniem piecem łukowym [179], gdzie implementacja systemu musi być poprzedzona jego analizą na wypadek pomyłki, która może nieść ze sobą poważne konsekwencje zarówno ekonomiczne jak i społeczne. Dlatego też jednym z głównych wyzwań stojących przed inteligencją obliczeniową jest wydobywanie z danych wiedzy w sposób, który pozwoli człowiekowi na jej zrozumienie i interpretację. Problem ten wciąż jest zagadnieniem otwartym determinując zaangażowanie wielu naukowców.
Ogólnie znanych jest wiele dróg pozwalających na interpretację wyników uczenia systemu. Najbardziej typowym rozwiązaniem są różnego rodzaju systemy regułowe, zapisujące wydobytą wiedzę w postaci reguł *jeżeli ... to ....* Część badaczy

preferuje jednak inne sposoby reprezentacji wiedzy - bądź w postaci rozkładów prawdopodobieństwa, bądź w postaci analizy przypadków.
Przeprowadzone badania z zakresu psychologii poznawczej, jak i kognitywistyki wskazują, iż ludzki umysł w większym stopniu pracuje w oparciu o analizę podobieństw do znanych sytuacji, czyli jest systemem bazującym na podobieństwie (SBM), niż poprzez analizę reguł [140]. Naturalnym więc jest próba adaptacji tych obserwacji w procesie automatycznego pozyskiwania wiedzy. Rezultatem tego są systemy reguł bazujących na prototypach (ang. prototype based rules) zwane również regułami-P [51]. Reguły-P stanowią alternatywę do innych znanych metod reprezentacji wiedzy, jednocześnie pozwalając na łatwą transformację do postaci innych form reprezentacji wiedzy, jak reguły rozmyte.
Nadrzędną cechą wszystkich systemów pozyskiwania wiedzy jest łatwość globalnego zrozumienia procesu wnioskowania, czyli transparentności modelu. Innymi słowy pożądane jest by model opisany był najprostszą strukturą o minimalnej liczbie parametrów, gwarantując maksymalną dokładność i generalizację. Sprowadza się to do problemu brzytwy Ockhama, gdzie poszukiwany jest kompromis pomiędzy prostotą a dokładnością.
Dlatego też głównym celem pracy jest zbadanie możliwości budowy modeli służących do klasyfikacji danych, reprezentujących zdobytą wiedzę w postaci reguły-P, spełniających kryterium brzytwy Ockhama. Problem ten sprowadza się do poszukiwania minimalnego zbioru wektorów referencyjnych oraz minimalnego podzbioru cech. Dlatego też problemy selekcji oraz optymalizacji prototypów wraz z zagadnieniem redukcji wymiarowości są kluczowymi elementami rozprawy. W rozprawie poddane zostało również analizie zagadnienie optymalizacji parametrów procesu uczenia oraz selekcja i dobór optymalnych algorytmów do konkretnego problemu.
Rozdział 2 definiuje pojęcie klasyfikacji oraz opisuje różne powszechnie znane metody wydobywania informacji z danych. W rozdziale 3 wskazano cel oraz zakres prac podjętych w ramach rozprawy. W kolejnym rozdziale 4, znajduje się opis systemu reguł prototypowych wraz z informacjami o ich budowie. Porównanie różnych miar odległości, włączając w to miary dla różnych typów atrybutów (ciągłych, symbolicznych oraz binarnych) wraz z miarami heterogenicznymi opisano w rozdziale 5. Zagadnienie selekcji prototypów oraz optymalizacji ich liczby (wraz z opisem autorskich rozwiązań) zostało opisane w rozdziałach 6 oraz 7 odpowiednio dla reguł typu k-NNoraz prototypowych progowych. Rozdział 8 to studium selekcji cech oraz meta uczenia, czyli optymalizacji i selekcji modelu użytego do uczenia konkretnego zbioru danych. W rozdziale 9 poddano analizie zagadnienie porównania reguł-P oraz reguł rozmytych (reguł-F). Wyniki wydobywania reguł prototypowych z realnych zbiorów danych wraz z ich porównaniem z rezultatami uzyskanymi innymi metodami ekstrakcji reguł omówiono w rozdziale 10. Ostatni rozdział 11 to podsumowanie pracy oraz możliwości dalszych badań.

Rozdział 2

Problem klasyfikacji oraz indukcji wiedzy

2.1 Problem klasyfikacji oraz metody jej oceny

Zagadnienie klasyfikacji danych lub rozpoznawanie wzorców (ang. pattern recognition) są elementami szerokiej grupy problemów CI. Problemy te należą do zagadnień uczenia z nauczycielem i możemy w nich wyróżnić:

- **przestrzeń danych wejściowych** - reprezentowaną jako n wymiarowy wektor $\mathbf{x} = [x_1, x_2, ..., x_n]^T$, którego poszczególne składowe x_i nazywane są alternatywnie cechami lub atrybutami [1] i mogą być one typu liczbowego (np. liczb rzeczywistych - wówczas mówimy o cechach ciągłych), mogą przyjmować wartości dyskretne porządkowe - wówczas mowa jest o atrybutach dyskretnych lub też mogą przyjmować wartości nieuporządkowane, jakościowe (np. typu *pogoda* = {słoneczna, pochmurna, deszczowa, śnieżna}) zwane atrybutami symbolicznymi oraz binarne.

- **przestrzeń wyjściową** - zwykle jednowymiarową, nazywaną klasą, rozumianą jako pojedynczy atrybut symboliczny C, określający zbiór etykiet klas przypisanych do pojedynczego wektora tworzących obiekt $\mathbf{o} = [\mathbf{x}, c]$

- **model lub klasyfikator** - funkcję $\mathcal{M}(\cdot)$ realizującą odwzorowanie $\mathcal{M}(\mathbf{x}; \alpha) \Rightarrow C$, gdzie α to zbiór parametrów modelu

Proces uczenia klasyfikatora $\mathcal{M}(\cdot)$ polega na adaptacji jego parametrów α w celu minimalizacji określonej funkcji błędu. Proces ten realizowany jest na podstawie zbioru treningowego $\mathbf{T} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_m]$, składającego się z m wektorów zwanych również obiektami lub instancjami. Podstawą procesu uczenia modelu jest założenie, że wszystkie wektory wejściowe prezentowane klasyfikatorowi pochodzą ze zbiorów o tych samych rozkładach prawdopodobieństwa $p(\mathbf{x}|C)$, przy czym poszczególne wektory zbioru treningowego losowane są w sposób niezależny.
Do oceny jakości modelu służy zwykle zbiór testowy, a proces ten nazywany jest testowaniem klasyfikatora. W procesie testowania modelowi prezentowany jest zbiór testowy, a jego zadaniem jest wyznaczenie wartości wyjściowych, czyli etykiet klas dla

[1] Niektórzy autorzy rozróżniają pojęcie cechy oraz atrybutu, jednakże w niniejszej pracy obydwa sformułowania stosowane są alternatywnie

		Rzeczywiste	
		Prawda	Fałsz
Oszacowane	Pozytywne	TP	FP
	Negatywne	FN	TN

Tablica 2.1: Macierz konfuzji

wszystkich elementów tego zbioru. Czasem w procesie uczenia używa się pojęcia zbioru walidacyjnego, będącego podzbiorem zbioru treningowego jednakże nie używanym w procesie uczenia. Służy on jedynie ocenie jakości klasyfikacji.
Zagadnienie oceny jakości działania klasyfikatora zwykle sprowadza się do wyznaczenia współczynników określających dokładność lub błąd klasyfikacji. W zależności od wymagań konstruktora wyróżnia się różne definicje dokładności i błędu klasyfikacji. Najpowszechniej używanym współczynnikiem oceny jest błąd Err lub dokładność Acc (2.1).

$$\begin{aligned} Err &= \frac{m^{err}}{m} \\ Acc &= 1 - Err \end{aligned} \tag{2.1}$$

gdzie m^{err} to liczba błędnie sklasyfikowanych wektorów zbioru testowego lub walidacyjnego, a m to całkowita liczba wektorów tego zbioru. Tak zdefiniowany błąd ma jednak pewną wadę, a mianowicie zafałszowuje zdolność klasyfikacji w przypadku zbiorów niezbalansowanych, czyli o różnej liczbie wektorów przypadających na poszczególne klasy. Przykładem tego może być sytuacja, w której mając zbiór treningowy o $m = 1000$ wektorach, z których $m_1 = 980$ pochodzi z klasy C_1, a $m_2 = 20$ z klasy C_1, klasyfikator większościowy klasyfikujący wszystkie wektory do klasy C_1 będzie charakteryzował się dokładnością rzędu 98%. Dlatego też coraz bardziej powszechnie używanym współczynnikiem oceny jest błąd zbalansowany ($BErr$) lub dokładność zbalansowana ($BAcc$) liczona jako średni błąd lub dokładność klasyfikacji poszczególnych klas (2.2)

$$\begin{aligned} BErr &= \frac{1}{c} \sum_{i=1}^{c} \left(\frac{m_i^{err}}{m_i} \right) \\ BAcc &= 1 - BErr \end{aligned} \tag{2.2}$$

Współczynnik ten znacznie wierniej oddaje realną dokładność klasyfikacji, traktując wszystkie klasy równomiernie, niezależnie od wewnętrznego rozkładu liczby wektorów w poszczególnych klasach.
Dwa przedstawione dotychczas parametry oceny klasyfikatora cieszą się dużą popularnością, jednak również współczynnik $BAcc$ nie jest pozbawiony wad. Niestety nie uwzględnia on wariancji jakości klasyfikacji poszczególnych klas, dlatego też jakość modelu $\mathcal{M}_1$ o dokładności klasyfikacji klas odpowiednio $C_1 = 90\%$ i $C_2 = 60\%$ oraz modelu $\mathcal{M}_2$ o dokładnościach $C_1 = 77\%$ i $C_2 = 73\%$ będzie miała taki sam ostateczny wynik $BAcc = 75\%$, co może być bardzo niepożądane. Dlatego też często używa się rożnych innych metod oceny, między innymi wyniki klasyfikacji często prezentowane są w postaci macierzy konfuzji (2.1).

Dla macierzy konfuzji pożądanym jest by elementy znajdujące się poza główną przekątną dążyły do zera. Definiuje się w niej parametry:

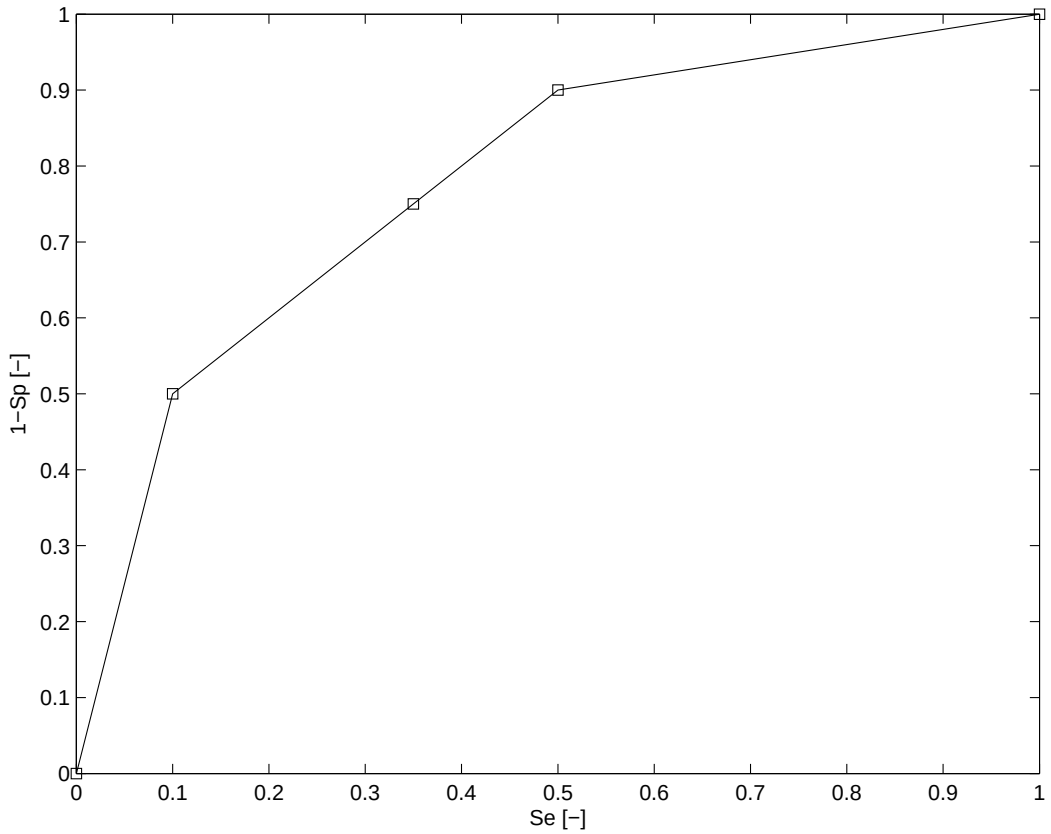
- parametr (ang. true positive) TP
- parametr (ang. false positive) FP

- parametr (ang. true negative) TN
- parametr (ang. false negative) FN
- parametr (ang. positive) $Pos = TP + FP$
- parametr (ang. negative) $Neg = TN + FN$
- parametr (ang. true) $True = TP + TN$
- parametr (ang. false) $False = FP + FN$

które pozwalają na wyznaczenie większości powszechnie używanych wskaźników, takich jak:

- Czułość lub wrażliwość (ang. sensitivity, recall) $Se = \frac{TP}{TP+FN}$
- Znamienność (ang. specificity) $Sp = \frac{TN}{TN+FP}$
- Precyzja (ang. precision) $P = \frac{TP}{TP+FP}$
- Miara F_β (ang. F_β measure) $F_\beta = \frac{(\beta^2+1)TP}{(\beta^2+1)TP+FP+\beta^2 FN}$, która dla $\beta = 1$ przyjmuje postać $F_{1.0} = \frac{2TP}{2TP+FP+FN} = \frac{2SeP}{Se+P}$
- Dokładność $Acc = \frac{TP+TN}{Pos+Neg}$
- Dokładność zbalansowana $BAcc = \frac{1}{c}\left(\frac{TP}{Pos} + \frac{TN}{Neg}\right)$

Każda z przedstawionych tutaj miar znalazła specyficzną dla siebie grupę zastosowań. W zagadnieniach medycznych szczególnym powodzeniem cieszą się miary czułości oraz znamienności wskazując jakość klasyfikacji poszczególnych klas, gdzie koszt popełnienia błędu może być różny dla różnych klas. Przykładem tego może być analiza sytuacji sklasyfikowania osoby chorej jako zdrowej oraz zdrowej jako chorej, gdzie koszt pierwszej pomyłki jest dużo większy niż drugiej. W zagadnieniach *odnajdywania informacji* (ang. information retrieval, IR) bardzo dużą popularność zyskały miara F_β oraz precyzja i wrażliwość czego dowodzą prace [110, 169].
W większości wypadków, gdy niezbędne jest określenie dopuszczalnego błędu klasyfikacji poszczególnych klas korzysta się z charakterystyki ROC (ang. receiver operating characteristic) [141]. Krzywą ROC wykreśla się jako zależność wrażliwości (Se) w funkcji dopełnienia znamienności (1-Sp) dzięki czemu możliwe jest wyznaczenie optymalnego progu klasyfikacji poszczególnych klas. Przedstawia to rys.(2.1). Na podstawie krzywej ROC wyznacza się również miarę jaką jest pole pod krzywą, zwaną współczynnikiem AUC (ang. area under curve). AUC informuje, który z klasyfikatorów lub rodzina klasyfikatorów jest najdokładniejsza spośród wszystkich zbadanych. Im większe pole pod krzywą, tym mamy do czynienia z lepszymi klasyfikatorami. W literaturze znane są również inne metody oceny dokładności klasyfikacji (jak np. opisane w pracy [40]). Różnorodność podejść wynika z faktu, iż nie można wskazać jednego uniwersalnego wskaźnika jakości klasyfikatora. Rozpoczęcie procesu uczenia modelu powinna więc poprzedzić analiza problemu oraz określenie zadań, jakie powinien on spełnić. W rozprawie do oceny jakości modeli posłużono się miarą *Acc* ze względu na powszechność metody oraz dostępność wyników porównawczych. Należy jednak zwrócić uwagę na fakt, iż coraz częściej używanym wskaźnikiem jakości klasyfikacji staje



Rysunek 2.1: Przykładowa charakterystyka ROC

się dokładność zbalansowana, czego dowodem może być sposób oceniania wyników w różnych konkursach analizy danych [77, 128, 172].
Przedstawione dotychczas współczynniki służą ocenie dokładności klasyfikacji, jednak równie ważnym parametrem klasyfikatora jest jego zdolność generalizacji [173]. Generalizacja rozumiana jest jako umiejętność uogólnienia zdobytej wiedzy, czyli miara informująca o możliwościach klasyfikacji danych do tej pory nie prezentowanych budowanemu modelowi. W literaturze znanych jest wiele metod pozwalających ocenić możliwości generalizacji modelu, tym samym wybrać model który najlepiej opisuje dane zagadnienie [61]. Jednym z typowych rozwiązań analizy generalizacji modelu jest zasada *minimalizacji długości opisu* (ang. minimum description length, MDL), która bazuje na analizie rozmiaru zbioru oraz stopnia złożoności modelu h wyrażonego w bitach (2.3).

$$h_{MDL} = \arg\min_{h \in H}(L_C(h) + L_C(\mathbf{X}|h)) \tag{2.3}$$

gdzie $L_C(h)$ jest długością opisu modelu h pod warunkiem kodowania C, natomiast $L_C(\mathbf{X}|h)$ jest długością opisu zbioru $\mathbf{X}$ pod warunkiem modelu h oraz kodowania C. Metoda MDL znalazła bardzo szerokie zastosowanie w algorytmach ekstrakcji reguł, gdzie między innymi jest stosowana do określenia rozmiaru drzew decyzji. Innym przykładem miary generalizacji jest miara Vapnika-Chervonenkisa zwany również wskaźnikiem VC oraz zasada minimalizacji ryzyka strukturalnego (ang. structural risk minimization, SRM) [170] zaproponowana przez Vapnika. Idea współczynnika SRM polega na uwzględnieniu obok dokładności modelu h drugiego czynnika opisującego

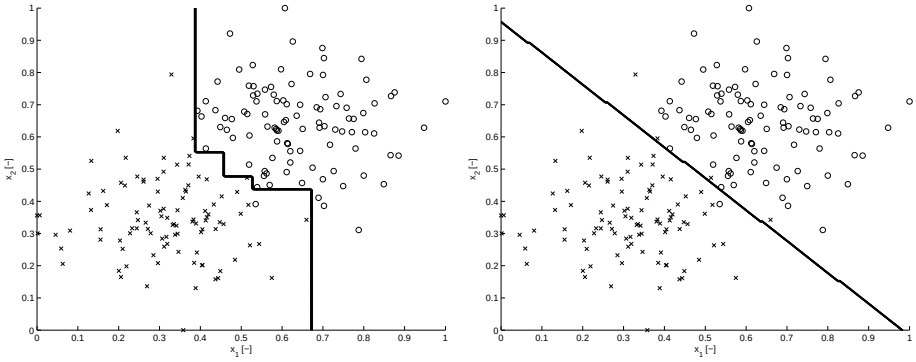
złożoność, tym samym wariancję modelu (2.4).

$$R(h) \leq R_{emp}(h) + \sqrt{\left(\frac{VC(h)(log(2m/VC(h)) + 1) - log(\eta/4)}{h}\right)} \tag{2.4}$$

Gdzie $R(h)$ to rzeczywisty koszt popełnienia błędu, $R_{emp}(h)$ to empiryczna wartość tego kosztu - błąd klasyfikacji, VC(h) - to wartość współczynnika Vapnika-Czervonenkisa, natomiast η to pewna stała. Inną bardzo powszechną metodą oceny jest algorytm walidacji krzyżowej oraz test pozostaw jeden (ang. leave one out). Dokładniejszy opis można znaleźć w rozdziale 8.2.

2.2 Formy reprezentacji wiedzy w problemach klasyfikacji danych

Proces uczenia modelu sprowadza się do pozyskiwania przez niego wiedzy na temat zagadnienia, które zostaje mu przedstawione. Cała idea uczenia polega zatem na dostrojeniu parametrów modelu α w taki sposób, aby jak najdokładniej odzwierciedlić charakterystykę zjawisk zachodzących w źródle jakim jest zbiór treningowy.
W zależności od budowy modelu mówi się o systemach, w których zgromadzona wiedza wyrażana jest *explicite* oraz *implicite*. Modele explicite przedstawiają wiedzę w postaci bezpośrednio zrozumiałej dla człowieka. Do tych grup metod można zaliczyć systemy klasycznych reguł dwuwartościowych, metody probabilistyczne, systemy bazujące na regułach rozmytych oraz systemy wnioskowania w oparciu o przypadki (ang. case based reasoning, CBR). Natomiast druga grupa klasyfikatorów ze względu na swoją budowę nie pozwala na interpretację zgromadzonej wiedzy przez człowieka, mówi się wówczas o tak zwanych czarnych skrzynkach (ang. black box), które potrafią przeprowadzić procedurę wnioskowania jednakże człowiek pozbawiony jest możliwości jej weryfikacji. Do tego typu modeli należą między innymi perceptron wielowarstwowy (MLP) czy klasyfikatory typu SVM. Próba wydobycia wiedzy z tego typu modeli sprowadza się do dopasowania modelu typu explicite do istniejącej czarnej skrzynki [130].
Każdy z klasyfikatorów explicite ma pewne charakterystyczne właściwości, które mają szczególne uzasadnienie dla różnych problemów. Jedną z częstych form reprezentacji wiedzy jest jej przedstawienie w postaci rozkładów prawdopodobieństw poszczególnych klas. Uczenie systemu sprowadza się do estymacji rozkładów prawdopodobieństwa a posteriori $p(C_i|\mathbf{x})$ oraz reguły wnioskowania *maksymalnej wartości a posteriori* (ang. maksimum a posteriori, MAP). Zaletą zapisu wiedzy w postaci rozkładów prawdopodobieństwa estymowanych z danych jest możliwość generacji nowych przypadków na podstawie tych rozkładów [168]. Słabością metod probabilistycznych jest ograniczona możliwość zrozumienia kształtu funkcji rozkładu prawdopodobieństwa w szczególności, gdy jest on skomplikowany i poszczególne cechy są wzajemnie skorelowane. Właściwości generacyjnej niestety pozbawione są systemy regułowe, które wydobytą wiedzą przedstawiają w postaci struktury typu: *jeżeli ... to ...* [56]. Cechą charakterystyczną takich systemów jest duża łatwość interpretacji wyników, gdyż w naturalny sposób jesteśmy przyzwyczajeni do analizy tego typu systemów, w szczególności bazujących na logice dwuwartościowej. Niestety rozwiązanie to posiada również wady. Przykładem tego jest problem klasyfikacji dwóch rozkładów gaussowskich w $\Re^2$ co przedstawia rys.(2.2) Optymalną granicą decyzyjną,



(a) Rozwiązanie z wykorzystaniem reguł klasycznych

(b) Rozwiązanie z wykorzystaniem reguł rozmytych oraz prototypowych

Rysunek 2.2: Przykładowy dwuklasowy problem klasyfikacji

gwarantującą maksymalną generalizację takiego systemu jest liniowa funkcja decyzyjna $\mathcal{M}(\mathbf{x}) = ax_1 + bx_2 + c$. Uzyskanie takiego rozwiązania przez systemy reguł twardych jest okupione dużą liczbą reguł, co ogranicza zdolność interpretacji takiego systemy, zaś zmniejszenie liczby reguł zmniejsza generalizację modelu. Tak zdefiniowany problem daje się jednak zapisać w prosty sposób jako dwie reguły typu rozmytego (refuły-F) lub prototypowego (reguły-P).

Reguły rozmyte stanowią swego rodzaju uogólnienie reguł twardych, wprowadzając logikę rozmytą, czyli logikę wielowartościową opisującą stopień przynależności obiektu do danego zbioru. Jednym z pierwszych zastosowań systemów rozmytych był problem uczenia modelu na zasadzie analizy wiedzy eksperta i jej wykorzystanie do klasyfikacji przypadków [61]. Późniejszy rozwój metod sztucznych sieci neuronowych doprowadził do integracji sieci neuronowych z systemami rozmytymi, co pozwoliło na automatyczne dostrajanie modelu wstępnie nauczonego przez eksperta. Również metody zbiorów rozmytych posiadają szereg wad. Między innymi narzucają one założenie niezależności atrybutów tak, iż funkcje przynależności definiowane są niezależnie dla każdej z cech i łączone rozmytymi operatorami przecięcia zbiorów. Jak pokazano w rozdziale 9 jedną z możliwych form interpretacji przesłanek reguł rozmytych jest podobieństwo do wzorców. Jednak zachodząca tutaj relacja nie jest symetryczna, gdyż duża grupa powszechnie stosowanych radialnych funkcji bazowych (sieci RBF) nie posiada właściwości separowalności. Przy czym separowalność definiowana jest jako możliwość dekompozycji funkcji bazowej na składowe dla poszczególnych atrybutów (jedyną separowalną radialną funkcją bazową jest funkcja Gaussowska).

Innym przykładem klasy problemów, który nie daje się rozwiązać za pomocą omówionych powyżej form reprezentacji wiedzy są zagadnienia klasyfikacji typu większościowe głosowanie (większość jest za). Problem ten rozwiązują reguły typu *M-z-N* (ang. M-of-N) [5], które działają na zasadzie analizy liczebności przypadków. Jeżeli co najmniej M pewnych cech ze zbioru N możliwych jest pozytywna wówczas reguła jest spełniona. Przykładem takiej reguły jest pojedynczy perceptron realizujący sumę poszczególnych składowych. Problem głosowania rozwiązują również omawiane w rozprawie reguły prototypowe progowe, określając odpowiedni próg aktywacji

$D(0, \sigma(\mathbf{wx})) > \Theta$.
Każda z wymienionych grup metod ma wielu przedstawicieli, a jednocześnie są one ciągle rozwijane, pozwalając na polepszenie jakości uzyskiwanych wyników, maksymalizację generalizacji oraz zwiększenie prostoty opisu zgromadzonej wiedzy. Poniżej przedstawione zostaną jedynie krótkie opisy wybranych najpopularniejszych spośród powszechnie stosowanych metod.

2.2.1 Metody bezpośredniej indukcji klasycznych reguł ostrych

Ta grupa metod charakteryzuje się wydobywaniem reguł ostrych bezpośrednio z danych. Typowym mechanizmem używanym do wydobywania reguł w tej grupie są metody przeszukiwania połączone z sekwencyjnym pokrywaniem przestrzeni wejściowej. Proces tworzenia reguł rozpoczyna się zwykle od najbardziej ogólnych - pokrywających maksymalną liczbę przypadków, a kończy na najbardziej szczegółowych - pokrywających małą liczbę wektorów.

Algorytm AQ

AQ [28] jest w rzeczywistości rodziną różnych metod bazujących na tym samym algorytmie podstawowym, modyfikując go poprzez dodanie różnych rozszerzeń lub funkcjonalności. Wersja podstawowa algorytmu AQ bazuje na wykorzystaniu metod przeszukiwania w sekwencyjnym pokrywaniu przestrzeni wejściowej regułami. Do definicji działania algorytmu Michalski zaproponował specjalne słownictwo używane powszechnie również w innych metodach sekwencyjnego pokrywania. Wprowadzonymi koncepcjami są: selektor - definiujący część warunkową reguły dla pojedynczego atrybutu, kompleks będący zbiorem selektorów dla różnych atrybutów oraz gwiazda stanowiąca zbiór selektorów, spośród których wybierany jest poprzednik reguły. Podstawową heurystyką używaną w algorytmie AQ przy generacji kompleksów jest „maksymalizacja liczby pozytywnych przypadków pokrywanych przez regułę przy wyłączeniu przypadków negatywnych". Przypadki pozytywne to zbiór wektorów należących do klasy C^+ dla których reguła jest tworzona, zaś przypadki negatywne C^- to zbiór wektorów nie należących do klasy C^+, $C^- = C - C^+$. Podstawowa wersja algorytmu była zdefiniowana dla atrybutów dyskretnych przy założeniu separacji klas. Te dwa założenia spowodowały duże ograniczenia stosowania algorytmu AQ, w szczególności wpłynęło to na niską generalizację uzyskanych reguł. Dlatego też późniejsze modyfikacje uwzględniały różne metody oczyszczania poprzez wstępne lub końcowe przetwarzanie danych jak w AQ11 [119] i AQ15 [118].

Algorytm CN2

Algorytm CN2 stanowi rozszerzenie algorytmu AQ, gdzie Clark i Niblett [29] analizowali sytuację ekstrakcji reguł z danych przy założeniu istnienia szumu. Główną różnicą jest tutaj rezygnacja z dotychczasowego kryterium indukcji reguł na rzecz miary entropii (2.5)

$$Q_E(q) = -\sum_{i=1}^{c} p(C_i|q) \log_2 p(C_i|q) \tag{2.5}$$

gdzie $p(C_i|q)$ jest względną liczbą wektorów z klasy C_i pokrytych przez dany kompleks q. Dzięki takiej modyfikacji możliwe staje się pokrywanie poprzez regułę niewielkiej ilości przypadków negatywnych, co znacznie wpływa na zdolności generalizacyjne algorytmu.

Kolejną modyfikacją w stosunku do algorytmu bazowego jest wbudowany algorytm oczyszczania reguł, podobny do stosowanego w systemach drzew decyzji. Pozwala on na usunięcie nieużytecznych reguł, które są efektem przetrenowania systemu. Ponieważ uzyskane w wyniku działania algorytmu reguły mogą się nakładać, rezultatem jest lista reguł czytana od najbardziej szczegółowej do najogólniejszej. Cecha ta wraz z algorytmem oczyszczania wymusiła również wprowadzenie do reguł dodatkowego warunku *jeśli nie* (ang. else), stosowanego w przypadku nie spełnienia żadnej z reguł. Algorytm CN2 również doczekał się wielu usprawnień podnoszących jego możliwości, czego przykładem może być praca [30]

Tabela decyzji

Tabela decyzji (ang. decision table) jest algorytmem ekstrakcji reguł zaproponowanym przez Kohaviego w [97]. W zaproponowanej metodzie autor posłużył się regułą *najistotniejszy z tabeli decyzji* (ang. decision table majority) do podejmowania arbitrażu. Bazuje ona na podjęciu decyzji w oparciu o największą częstotliwość występowania danej wartości w zbiorze zapamiętanych wzorców, która jest zgodna z danym wektorem testowym.
W zaproponowanym podejściu autor określa dwa elementy składowe

- **schemat** (ang. schema), który stanowi zbiór cech
- **ciało** (ang. body), które stanowi zbiór etykietowanych wektorów mających określoną wartość dla każdej z cech zawartych w **schemacie**

W wersji bazowej algorytmu, jedynie podzbiór cech jest optymalizowany stosując metodę opakowywania (ang. wrappers approach) w selekcji cech.

2.2.2 Drzewa decyzji

Drzewa decyzji to bogata grupa metod, która w odróżnieniu od algorytmów bezpośredniej indukcji reguł, stosuje strukturę drzewa, pokrywającą przypadki z przestrzeni wejściowej stosując różne kryteria statystyczne. Najczęstszym efektem działania drzew decyzji jest drzewiasta struktura *klasycznych reguł*. Znane są jednak implementacje stosowane do budowy alternatywnych typów reguł, jak zaproponowany w pracy Ichihashi i inni [76] algorytm rozmytych drzew.
Wynik działania drzew decyzji daje się zwykle przedstawić w postaci klasycznych liniowych zbiorów reguł. Często bywa to jednak okupione zmianą kształtu granicy decyzyjnej, co powoduje, że obydwa zbiory reguł o strukturze liniowej oraz drzewiastej nie są sobie równoważne. Przykładem tego jest algorytm C4.5-Rules.

Algorytm CART

Metoda CART (ang. classification and regression trees) [22] jest jednym z najpopularniejszych i najdokładniejszych algorytmów drzew decyzji. W drzewie tym zastosowano binarny podział węzłów, tak że z każdego węzła wychodzą dwie gałęzie zakończone kolejnymi podwęzłami. Dla każdego węzła wyznaczana jest etykieta klasy jako najbardziej liczna reprezentacja wektorów z danej klasy. W CART zastosowano różne kryteria statystyczne określające czystość węzłów. W większości bazują one na różnego rodzaju entropiach. Autorzy sugerują używanie indeksu Gini (2.6), który jest

szczególnym przypadkiem entropii Renyiego (2.7) ze współczynnikiem $w = 2$ (przy czym operator *log* może tutaj zostać pominięty ze względu na właściwość porównywania wartości indeksu).

$$Q_G(q) = 1 - \sum_{i=1}^{c} p(C_i|q)^2 \tag{2.6}$$

$$Q_R(q) = \frac{1}{1-w} \ln \left(\sum_{i=1}^{c} p(C_i|q)^w \right) \tag{2.7}$$

gdzie q oznacza węzeł, dla którego liczone jest kryterium.
Innym często spotykanym kryterium jest miara entropii Shannona opisana zależnością (2.5). Autorzy zaproponowali również metodę pozwalającą na przetwarzanie danych niekompletnych, poprzez wyznaczenie dla każdego węzła alternatywnych atrybutów, analizowanych gdy aktualna wartość najlepszego atrybutu użytego w węźle jest niedostępna.

Algorytm ID3

ID3 [143] jest drzewem decyzji bazującym na indeksie wywodzącym się z teorii informacji zwanym *współczynnikiem przyrostu informacji* (ang. information gain) (2.8)

$$Q_{IG}(q, f) = E(q) - \sum_{v \in U(f)} \frac{|q_v|}{|q|} E(q_v) \tag{2.8}$$

gdzie $U(f)$ - jest zbiorem wszystkich możliwych unikatowych wartości atrybutu f, q_v jest podzbiorem q dla którego cecha f przyjmuje wartości v, zaś $|q|$, $|q_v|$ to odpowiednio liczebność q i q_v. W odróżnieniu od CART, algorytm ID3 nie jest drzewem binarnym, lecz liczba podwęzłów jest równa liczbie unikalnych wartości cechy $U(f)$. Wadą algorytmu ID3 jest praca jedynie na atrybutach dyskretnych lub symbolicznych, co znacznie ograniczyło praktyczne możliwości wykorzystania tej metody.

Algorytm C4.5

C4.5 [143] jest modyfikacją algorytmu bazowego, którym był algorytm ID3. W omawianej metodzie Quinlan przede wszystkim dodał nowe funkcjonalności usuwające słabości algorytmu bazowego:

- zmodyfikował funkcję kryterium, zwaną stosunkiem zysku informacyjnego (ang. information gain ratio) (2.9)
- zaproponował wsparcie dla cech ciągłych
- umożliwił pracę z brakującymi wartościami
- zmodyfikował metodę oczyszczania, by zwiększyć zdolność generalizacji algorytmu

$$Q_{IGR}(q, f) = \frac{Q_{IG}(q, f)}{E(q)} \tag{2.9}$$

W C4.5 atrybuty ciągłe w odróżnieniu od dyskretnych dzielone są na dwa podwęzły (jak w drzewach binarnych), gdzie pomiędzy wartościami cechy minimalizowane jest

kryterium (2.9) podziału na dwie części. Jako kryterium oczyszczania drzewa została użyta metoda statystyczna mierząca istotność różnicy pomiędzy danym węzłem a jego podwęzłami.
Praktyka pokazała że obok CART algorytm C4.5 jest jedną z najdokładniejszych i najczęściej stosowanych implementacji drzew decyzji.

Algorytm SSV

SSV (ang. separability split value) [65] jest bardzo podobną metodą konstrukcji drzewa do algorytmu CART (jest również drzewem binarnym). Główne różnice dotyczą funkcji kryterium zwanej SSV (2.10) oraz metody oczyszczania.

$$\begin{gathered} SSV(s,q,f) = 2\sum_{i=1}^{c} |LS(s,f,q_{C_i})||RS(s,f,q_{C\neq C_i})| - \\ \min\left(|LS(s,f,q_{C_i})|, |RS(s,f,q_{C_i})\right) \end{gathered} \tag{2.10}$$

gdzie

$$\begin{aligned} LS(s,f,q) &= \begin{cases} x \in q, f(x) < s & \text{jeśli q jest ciągłe} \\ x \in q, f(x) \in s & \text{w przeciwnym wypadku} \end{cases} \\ RS(s,f,q) &= f \setminus LS(s,q,f) \end{aligned} \tag{2.11}$$

Gdzie f to cecha dla której wartość kryterium jest wyznaczana, s - to wartość podziału, natomiast q to zbiór wartości dla których indeks jest liczony.
Użyta w SSV metoda oczyszczania wykorzystuje algorytm walidacji krzyżowej (ang. cross validation) do wyznaczenia optymalnego progu przycięcia. W procesie uczenia budowane jest kompletne drzewo, a następnie poprzez statystykę wyznaczoną w walidacji krzyżowej określany jest optymalny rozmiar drzewa gwarantujący maksymalną zdolność generalizacji. Dzięki tym zabiegom doświadczenia na realnych zbiorach danych pokazały dużą skuteczność opisywanego algorytmu.

2.2.3 Sieci neuronowe a systemy regułowe

Sieci neuronowe to grupa metod uczenia, której inspiracją było odwzorowanie budowy neuronów znajdujących się w ludzkich mózgach. Dlatego też przy ich konstrukcji nie uwzględniano możliwości interpretacji uzyskanych wyników uczenia przez ludzi, a w szczególności ich zapisu jako zbioru reguł. Spowodowało to, że duża grupa metod regułowych wywodzących się z sieci neuronowych bazuje na próbie odtworzenia wiedzy zgromadzonej w sieciach, czyli należy do grupy metod indukujących reguły z nauczonych modeli. Istnieją jednak sieci dedykowane indukcji reguł, których konstrukcja pozwala na bezpośrednią interpretację regułową. Tego typu sieci bazują często na strukturze perceptronu wielowarstwowego (MLP), których neurony mają schodkową funkcję aktywacji dzięki czemu pojedynczy neuron może być interpretowany jako koniunkcja przesłanek stanowiących wejście neuronu. Przykładem takiego algorytmu jest Search-MLP [1, 55, 99]. W literaturze można jednak spotkać również inne podejścia do zagadnienia wydobywania reguł z sieci neuronowych, czego przykładem są opisane poniżej metody.

Algorytm TREPAN

Trepan stanowi połączenie sieci neuronowych z algorytmami drzew decyzji. W metodzie ekstrakcji reguł zaproponowanej przez Craven i Shavlik [34] sieć neuronowa

traktowana jest jako wyrocznia, pozwalająca na etykietowanie nowo generowanych wektorów treningowych. Proces ekstrakcji reguł polega na nauczeniu drzewa decyzji w oparciu o wyniki uzyskane zarówno ze zbioru treningowego, jak i poprzez jego rozszerzenie uzyskane w wyniku losowania nowych wektorów treningowych, dla których sieć neuronowa przypisuje odpowiednie etykiety klas. Podczas konstrukcji drzewa zakłada się, że podział węzła następuje dopiero po przeanalizowaniu dużej liczby przypadków (ok. 10^3). Dzięki takiemu zabiegowi algorytm ten staje się dużo bardziej stabilny i dokładny niż klasyczne drzewa decyzji, w których problemem jest mała liczba wektorów przypadająca na węzeł w dolnej części drzewa.
Dużym atutem algorytmu TREPAN jest możliwość wykorzystania go do ekstrakcji reguł z niemal każdej metody pozbawionej naturalnej interpretacji regułowej.

Algorytm NeuroRule oraz M-of-N3

Algorytmy NeuroRule [155] oraz M-of-N3 [154] są metodami, które odtwarzają wiedzę zapisaną w sieci neuronowej przy czym M-of-N3 dokonuje ekstrakcji reguł typu M-z-N. Warunki budowy sieci określają, binarną postać atrybutów wejściowych, oraz konstrukcję sieci składającą się z trzech warstw: wejściowej, ukrytej oraz wyjściowej, przy czym neurony warstwy ukrytej mają funkcję aktywacji typu $\tanh(K_1, w)$ oraz dla algorytu M-of-N3 spełniają warunek (2.12)

$$|\tanh(K_1, w)| \approx 1 \tag{2.12}$$

Jeżeli warunek ten nie jest spełniony uzyskiwany jest algorytm NeuroRule.
W sieci neuronowej zaimplementowano również algorytm oczyszczania usuwający połączenia o małej wartości wag. Proces oczyszczania sieci realizowany jest iteracyjnie, gdzie po każdorazowym usunięciu mało istotnych połączeń sieć jest douczana i proces oczyszczania ponownie uruchamiany, aż do momentu, w którym żadne z połączeń nie jest usuwane.
Proces pozyskiwania reguł można opisać jako:

1. Naucz i oczyść sieć neuronową
2. Grupuj wartości aktywacji każdego z neuronów warstwy ukrytej
3. Stwórz reguły klasyfikacji na podstawie zgrupowanych wartości aktywacji
4. Zamień warunki reguł stworzonych w kroku 3 przez warunki typu M-z-N

gdzie w kroku drugim proces klasteryzacji realizowany jest w oparciu o algorytm dyskretyzacji Chi2 lub ChiMarge [112]. Natomiast w trzecim kroku posłużono się algorytmem X2R w celu wydobycia klasycznych rozłącznych reguł binarnych, które w ostatnim kroku w przypadku algorytmu M-of-N3 zamieniane są reguły typu M-z-N.

Algorytm VIA

Algorytm VIA (ang. validity interval analysis) zaproponowany przez Thurana [165] służy do ekstrakcji reguł z sieci neuronowych uczonych na zasadzie wstecznej propagacji błędu. Bazuje on na zasadzie analizy wzbudzeń poszczególnych neuronów względem przypisanych interwałów, zwanych interwałami walidacyjnymi (ang. validity interval). Jeżeli aktywacja danych neuronów odpowiada interwałom, wówczas uznawane są one za

spójne i zapisywane w bazie wiedzy. Do weryfikacji interwałów używany jest algorytm wstecznej propagacji błędu, dzięki czemu interwał z wyjścia jest mapowany na wejście danego neuronu. Algorytm ten jest realizowany iteracyjnie, adaptując każdorazowo interwały walidacyjne oraz wyłączając interwały dotychczas niespójne.

Algorytm MLP2LN

Algorytm MLP2LN [57] jest dedykowanym algorytmem poszukiwania reguł bazującym na sieciach MLP. Autorzy - Duch, Adamczak, założyli specjalną strukturę sieci, w której na wejścia warstwy ukrytej podawane są odpowiednie interwały sygnałów wejściowych sieci, wstępnie przetworzone jako różnica dwóch funkcji sigmoidalnych. W procesie uczenia wykorzystali również fakt możliwości zmiany nachylenia skosu funkcji sigmoidalnej, co pozwala zarówno na wykorzystanie algorytmów gradientowych oraz propagacji wstecznej do optymalizacji wag sieci przy małym skosie funkcji sigmoidalnej oraz interpretację binarną przy skosie dążącym do inf. Dlatego też wraz z kolejnymi epokami uczenia sieci zmianie ulega również współczynnik skosu s. W celu poprawienia jakości uczenia modyfikacji poddana została również funkcja kosztu (2.13)

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p}\sum_{i}\left(y(\mathbf{x}_i,\mathbf{W}) - d_i^p\right)^2 + \frac{\lambda_1}{2}\sum_{i>j} w_{ij}^2 + \frac{\lambda_2}{2}\sum_{i>j} w_{ij}^2(w_{ij}+1)^2(w_{ij}-1)^2 \tag{2.13}$$

gdzie $y(\mathbf{x}_i, \mathbf{w})$ jest wyjściem sieci wyznaczonym dla wektora $\mathbf{x}_i$ oraz wag $\mathbf{w}$, d_i jest etykietą wektora $\mathbf{x}_i$, w_{ij} to i,j-ty element macierz wag, natomiast λ_1 oraz λ_2 to współczynniki definiowane przez użytkownika, rozszerzone o dwa dodatkowe czynniki. Odpowiednio drugi czynnik wpływa na stopień skomplikowania modelu - wymuszający małe wartości wag $\mathbf{w}$ (selekcję cech na wejściach neuronów) oraz trzeci wymusza duże wartości wag [-1,0,1]. Wagi poszczególnych czynników funkcji kosztu są zmieniane w procesie uczenia, tak by w końcowym etapie dominującą wartość miał ostatni czynnik odpowiedzialny za binaryzację wyjścia sieci.

2.2.4 Systemy neuronowo-rozmyte

Zbiory rozmyte oraz logika rozmyta stanowią podstawę reguł rozmytych oraz różnych innych metod zwanych ogólnie rozmytym obliczaniem (ang. soft computting) [138]. Reguły rozmyte można scharakteryzować jako uogólnienie tradycyjnych reguł logiki klasycznej, w których spełnienie poszczególnych przesłanek oraz konkluzji jest określane jako wartość w przedziale $[0, 1]$.
Tradycyjne podejście do budowy reguł rozmytych bazowało na wykorzystaniu wiedzy eksperta oraz naturalnej interpretacji lingwistycznej opisu zagadnienia [61]. Jednakże sukces sztucznych sieci neuronowych doprowadził do połączenia i zbudowania uniwersalnego narzędzia, pozwalającego na zautomatyzowanie procesu wydobywania wiedzy z danych w postaci reguł rozmytych oraz inicjalizacji procesu uczenia poprzez wbudowanie w system wiedzy eksperta. Zagadnienie to analizowane było między innymi w pracach Rutkowskiej i Rutkowskiego [145], oraz Czogały i Łęskiego [35]. Budowa sieci neuronowo-rozmytych pozwala na wyrażenie struktury modelu jako sieci neuronowej, która jednocześnie posiada interpretację jako zbiór reguł rozmytych. Praktyka pokazała, że to podejście stało się bardzo popularne czego wynikiem były różne metody budowy modeli typu neuronowo-rozmytych. Poniżej opisano kilka najbardziej znanych.

System ANFIS

ANFIS [80, 81] jest jednym z pierwszych modeli neuronowo-rozmytym. Podstawą jego konstrukcji było założenie, że jest to sieć jednokierunkowa o warstwach różniczkowalnych, co pozwoliło na wykorzystanie algorytmu propagacji wstecznej w procesie uczenia model. Założona przez Janga postać sieci składała się z pięciu warstw, którymi kolejno były: warstwa pierwsza - wyznaczająca wartości funkcji przynależności dla danego wejścia sieci, co odpowiada określeniu aktywacji poszczególnych przesłanek; warstwa druga realizowała koniunkcję przesłanek dla każdej z reguł; warstwa trzecia wyznaczała znormalizowaną wartość aktywacji przesłanki reguły względem pozostałych reguł; warstwa czwarta odpowiedzialna była za konkluzję reguły, wyznaczając poziom jej aktywacji; ostatnia piąta warstwa sumowała odpowiedzi różnych reguł wyznaczając ostateczną odpowiedź systemu. Konstrukcja sieci była równoważna modelom rozmytym Sugeno i Tsukamoto.
W modelu ANFIS inicjalizacja sieci była realizowana najczęściej bądź poprzez klasteryzację, bądź poprzez równomierny podział wartości atrybutów.

System NefClass

NefClass to algorytm ekstrakcji reguł rozmytych w procesie klasyfikacji wzorców zaproponowany przez Nauck i Kruse [127] i rozwijany również przez Rutkowską [144]. Proces uczenia systemu rozpoczyna się od inicjalizacji poprzez równomierny podział atrybutów na odpowiednie wartości lingwistyczne. Ze względu na łatwość interpretacji uzyskanych reguł autorzy posłużyli się operatorem agregacji maximum, co wpłynęło na sposób optymalizacji systemu. Ze względu na jego nieliniowość autorzy posłużyli się algorytmem pseudo-gradientowym. Cechą charakterystyczną modelu było dodanie procesu oczyszczania reguł poprzez 4 różne metody: korelację, częstość klasyfikacji, nadmiarowość i rozmycie [126]. Uzyskany w ten sposób rezultat zawierał mały zbiór reguł i przesłanek w poszczególnych regułach.

System FuNN

FuNN (ang. fuzzy neural network) [89] jest w rzeczywistości ogólną ideą budowy systemów neuronowo-rozmytych, której wynikiem jest zbiorem różnych metod pozwalających na uczenie sieci neuronowej, przedstawiającej swoją strukturą zbiór reguł rozmytych. Interpretacja poszczególnych warstw sieci jest zbliżona do modelu ANFIS i odpowiadają one odpowiednio: warstwa pierwsza to warstwa wejściowa, druga odpowiada przesłankom reguł, trzecia tworzy reguły, czwarta nazywana jest warstwą akcji, a piąta to warstwa wyjściowa zwracająca wynik działania sieci. FuNN zakłada różne metody uczenia sieci, w tym poprzez wsteczną propagację, algorytmy genetyczne oraz ewolucyjne. Część metod wchodzących w FuNN została opracowana do automatycznego procesu adaptacji zarówno parametrów, jak i typu użytych zbiorów rozmytych czy odpowiednich operatorów rozmytych [187]. Kolejnym etapem rozwoju sieci FuNN była idea Evolving FuNN, pozwalające m.in. na automatyczny dobór liczby reguł oraz uczenie inkrementacyjne (ang. on-line learning) bazujące na ogólnym procesie ramowym nazwanym ECOS (ang. evolving connectionist systemsn), dzięki czemu możliwe jest ciągłe douczanie modelu [90].

System FSM

FSM (ang. feature space mapping) [59, 54, 1] jest algorytmem neuronowo-rozmytym opracowanym przez zespół R. Adamczak, W. Duch oraz G. Diercksen. Architektura sieci FSM jest zbliżona zarówno do sieci radialnych (RBF), jak i konstruktywistycznej sieć RAN (ang. resource allocating network) oraz sieci do wydobywania reguł o nazwie RecBFN (ang. rectangular basis functions network). Funkcja decyzyjna sieci FSM jest typu (2.14).

$$FSM(\mathbf{x}) = C\left(\max_j \left(G_j(\mathbf{x}; \mathbf{p}, \sigma)\right)\right) \tag{2.14}$$

gdzie $\mathbf{x}$ jest wektorem wejściowym, $\mathbf{p}$ jest centrum funkcji radialnej, natomiast σ to parametr określający jej szerokość, a $C(\cdot)$ jest funkcją zwracającą etykietę klasy. Bazuje ona jednak na separowalnych funkcjach bazowych (2.15).

$$G(\mathbf{x}; \mathbf{p}, \sigma) = \prod_i G(x_i; p_i, \sigma_i) \tag{2.15}$$

Poprzez takie podejście uzyskano interpretację systemu jako zbioru reguł rozmytych, co powoduje, że algorytm ten można analizować jako sieć neuronowo-rozmytą.
Algorytm FSM należy do grupy metod konstruktywistycznych, dzięki czemu liczba funkcji podobieństwa uzyskiwana jest w sposób automatyczny. Główną różnicą sieci FSM w stosunku do metod z rodziny RBF jest szeroka gama różnych funkcji podobieństwa, w tym trójkątnej, trapezoidalnej, Gaussowskiej, bicentralnej, itp. Model FSM uczony jest w oparciu o algorytm zbliżony do sieci RAN, jednak autorzy dokonali w nim modyfikacji mających na celu usprawnienie działania systemu.

Rozdział 3

Cel i zakres pracy

Zagadnienie budowy systemów reguł prototypowych jest stosunkowo nową koncepcją. Znane dotychczas z literatury rozwiązania systemów redukcji liczby prototypów koncentrowały się głównie na zwiększeniu dokładności klasyfikacji klasyfikatora k-NN, nie były one natomiast używane w celu zrozumienia danych. Algorytmy te omówione w rozdziale 6, nie wyczerpują problemu, gdyż niezbędne są metody które poszukują kompromisu pomiędzy dokładnością klasyfikacji (zdolnością generalizacji) a złożonością modelu, gdyż jedynie taki kompromis pozwoli na zrozumienie zbudowanego w sposób automatyczny modelu. Analiza literatury wskazuje również na coraz większe zainteresowanie systemami bazującymi na podobieństwie. Opracowane przez Ducha podstawy teoretyczne takich systemów [49] wskazują na ich szeroką uniwersalność i możliwość alternatywnej reprezentacji wiedzy w nich zgromadzonej. W literaturze problematykę budowy systemów bazujących na prototypach bardzo szeroko analizowali m.in. Wettschereck, Dietrich, Aha, Martinez, czy też Skala, dając szerokie podstawy systemów redukcji liczby prototypów. Późniejsze prace Kunchevy oraz Bezdeka rozwinęły te koncepcje, czego wynikiem był *uogólniony klasyfikator najbliższego prototypu* [103]. Szereg nowych prac i koncepcji koncentruje się obecnie wokół Duina i Pękalskiej, którzy systematycznie badają systemy bazujące na podobieństwie.
Głównym celem budowy systemów reguł prototypowych jest konieczność znalezienia alternatywy w procesie drążenia danych w stosunku do systemów reguł rozmytych. Z jednej strony systemy rozmyte znalazły bardzo szerokie zastosowanie w różnych dziedzinach nauki, jednak zupełnie nowe możliwości daje spojrzenie na badany problem z punktu widzenia charakterystycznych wzorców zwanych prototypami. Dlatego też rozwiązaniem tak zdefiniowanego problemu mogą być systemy bazujące na prototypach, które zgromadzoną w nich wiedzę przedstawiają w postaci obiektów umiejscowionych w przestrzeni wejściowej budowanego modelu.
Ogólnie problematyka budowy systemów bazujących na podobieństwie, a w szczególności systemów reguł prototypowych jest bardzo obszerna i składają się na nią zarówno zagadnienia selekcji prototypów, jak również ich ważenia oraz kwestia selekcji i ważenia cech. Bardzo poważnym problemem jest dobór odpowiednich miar podobieństwa i odległości. Opracowane przez Salsberga i innych, a później rozwijane przez Wilsona i Martineza miary z rodziny VDM oraz miary heterogeniczne stanowią poważne ogniwo integrujące systemy bazujące na prototypach z problemem analizy danych o różnych typach cech.
Pojawia się jednak problem, jaka jest optymalna miara podobieństwa oraz jakich miar najlepiej jest używać, w jaki sposób dobierać oraz jak realizować problematykę selekcji cech. Ogólnie znane twierdzenie *nie ma obiadu za darmo* (ang. no free lunch) wskazuje,

iż jednoznaczna odpowiedź na tak zadane pytanie nie istnieje. Bogata liczba różnych dostępnych metod selekcji cech oraz miar odległości zmusza jednak do przeprowadzenia wstępnej weryfikacji skuteczności poszczególnych z nich i wyłonienia grupy, którą będzie się analizowało podczas budowy ostatecznego modelu. Dotyczy to szerokiego problemu również innych parametrów, jak i stopni przetwarzania danych, które składają się na ostateczny model, który zostanie wykorzystany do ostatecznej analizy.

Cel:
Głównym celem pracy jest zbadanie skuteczności różnych rozwiązań stosowanych powszechnie do budowy systemów bazujących na podobieństwie oraz ich adaptacja i zbadanie możliwości ich wykorzystania do ekstrakcji reguł prototypowych. W tym celu dokonano adaptacji i modyfikacji probabilistycznych miar odległości oraz zaproponowano trzy nowe miary, wywodzące się z metod nieparametrycznej estymacji funkcji rozkładów gęstości prawdopodobieństwa (GVDM, LVDM, PVDM). Wyniki tych badań zostały opublikowane w pracach [180, 178, 16]. Ponadto w rezultacie badań zaproponowano nową metodę selekcji wektorów prototypowych bazującą na metodzie warunkowego grupowania danych [17]. W tym celu wykorzystano algorytm CFCM oraz przeprowadzono jego integrację z algorytmem LVQ łącząc zalety obydwu rozwiązań. W rozprawie rozważano też zagadnienie optymalizacji liczby prototypów, czego wynikiem jest *algorytm wyścigu* [17] stanowiący alternatywę do metod przeszukiwania powszechnie stosowanych przez wielu badaczy.
Kolejnym celem realizowanej przez autora rozprawy, był nowy system ekstrakcji reguł prototypowych progowych. W tym celu zbudowano algorytm OPTDL [15, 19], który w odróżnieniu od systemów heterogenicznych drzew decyzji charakteryzuje się płaską listą reguł co często pozwala na ułatwienie interpretacji uzyskanych wyników.
W rozprawie poddano analizie problematykę integracji różnych metod selekcji cech w zastosowaniu do budowy systemów reguł prototypowych. Należy jednak zauważyć, że uzyskane w tej części wyniki mają szersze zastosowanie i nie dotyczą jedynie problemu systemu reguł prototypowych.
Ostatnim rozważanym w rozprawie zagadnieniem jest zależność pomiędzy systemami reguł prototypowych a systemami rozmytymi. W rozdziale 9 poddano analizie możliwości transformacji wiedzy zgromadzonej w jednych systemach w drugie oraz korzyści z tego tytułu wynikające.

Zakres:
W celu realizacji omówionych powyżej zadań opracowano nowe probabilistyczne miary odległości oraz przeprowadzono szereg testów porównawczych, dokonując weryfikacji ich przydatności. W tym celu zbadano wpływ doboru odpowiedniej metryki na maksymalizację dokładności klasyfikacji oraz generalizacji budowanego modelu, jak również poddano analizie możliwości interpretacji uzyskanych wyników.
Przebadano również problem optymalnej reprezentacji danych w postaci wektorów referencyjnych. Zadanie to zrealizowano proponując nowy algorytm selekcji prototypów bazujący na warunkowym grupowaniu danych (CFCM) oraz integrując go z algorytmem LVQ. Przydatność opracowanego algorytmu sprawdzono porównując uzyskane na realnych zbiorach danych wyniki z różnymi znanymi z literatury metodami selekcji i optymalizacji prototypów. W procesie testowania rozważano uzyskaną średnią dokładność klasyfikacji oraz wybraną liczbę wektorów referencyjnych.
W pracy omówiono również nowy algorytm selekcji prototypów dla reguł prototypowych

progowych OPTDL i poddano weryfikacji jego dokładność porównując ją z wyniki klasyfikacji uzyskanymi dla metod drzew heterogenicznych.
Zagadnienie doboru optymalnego podzbioru cech dokonano niezależnie dla dwóch różnych grup metod. Dla metod rankingowych dokonano weryfikacji skuteczności wskaźników bazujących głównie na teorii informacji poprzez ich przetestowanie na realnych, jak i na sztucznych zbiorach danych o znanej istotności poszczególnych cech. Natomiast metody selekcji bazujące na przeszukiwaniu oraz metodę selekcji w oparciu o drzewa decyzji zweryfikowano jedynie na realnych zbiorach danych.
Ostatecznie dokonano porównania integrując opisane powyżej metody selekcji cech z metodami selekcji prototypów z powszechnie znanymi algorytmami wydobywania reguł zarówno klasycznych jak i binarnych.
Analizę relacji pomiędzy systemami bazującymi na prototypach a systemami rozmytymi skoncentrowano na modelach prototypowych typu k-NNoraz modelu rozmytym TSK z singletonami w konkluzji oraz operatorem agregacji konkluzji typu max. Przeprowadzono również analizę relacji pomiędzy różnymi typami miar odległości oraz odpowiadającymi im operatorami iloczynu logicznego zbiorów rozmytych. Zbadano również relacje pomiędzy operatorami implikacji rozmytej a rozwiązaniami stosowanymi w systemach ważonego głosowania algorytmu k-NN.

Rozdział 4

Metody reguł prototypowych

4.1 Wstęp

Bardzo interesującym narzędziem analizy danych są metody bazujące na podobieństwie (ang. similarity based methods, SBM) [49, 45]. Ich głównym atutem, jak zostało to wspomniane w rozdziale 1, jest możliwość integracji wielu różnych typów systemów analizy danych, jak bazujące na prawdopodobieństwie, sieci neuronowe czy metody zbiorów rozmytych. Z tej grupy metod wywodzą się również systemy reguł prototypowych jako narzędzie pozwalające na reprezentację wiedzy w postaci pewnego zbioru reguł zrozumiałych dla człowieka.
Wszystkie metody wywodzące się z SBM reprezentują wiedzę w postaci zbioru prototypów (wektorów referencyjnych) oraz odpowiednich miar odległości wraz z modułem integrującym informację pochodzącą z poszczególnych wzorców. Systemy reguł prototypowych mają identyczną konstrukcję, jednakże nacisk położony jest w nich na możliwie najprostszą reprezentację wiedzy. Cel ten osiągany jest poprzez redukcję liczby wektorów referencyjnych (selekcję prototypów) oraz minimalizację liczby cech użytych do konstruowania modelu. Dzięki takiemu podejściu możliwa staje się reprezentacja wiedzy pochodzącej z dużych zbiorów danych w postaci zaledwie kilku prototypów. W wielu dziedzinach życia taki sposób jest niezmiernie przydatny i stanowi alternatywę w stosunku do innych form zapisu wiedzy jak systemy klasycznych reguł lub też reguł rozmytych. Przykładem zastosowań systemów reguł-P są problemy medyczne gdzie znalezienie w zbiorze danych kilku przypadków pozwalających na dyskryminację z dużą dokładnością może przynieść wiele korzyści. Dzięki temu możliwe staje się znalezienie charakterystycznych dla danych schorzeń symptomów wraz z ich odpowiednimi wartościami.
Cechą charakterystyczną reguł prototypowych jest ich uniwersalność. Pozwalają one na integrację różnych form wyrażenia reguł. Między innymi za ich pomocą możliwe jest zapisanie zarówno reguł klasycznych poprzez wykorzystanie odległości Czebyszewa, jak i reguł rozmytych - w postaci reguł prototypowych typu k-NNz separowalnymi funkcjami odległości oraz reguł typu M-z-N w postaci np. reguł progowych.
Reguły prototypowe charakteryzują się również bardzo dobrymi wynikami w realnych zastosowaniach. Przykładem tego mogą być wyniki uzyskane dla danych reprezentujących choroby raka piersi (zbiór Wisconsin Breast Cancer), gdzie zaledwie pojedyncza reguła pozwala na klasyfikację ze średnią dokładnością 97.3%.
W systemach reguł-P można wyodrębnić dwa podstawowe typy reguł różniące się między sobą sposobem interpretacji. Wyróżnia się więc reguły najbliższego sąsiada lub k-

najbliższych sąsiadów (ang. k-nearest neighbor rule, k-NN) oraz *reguły prototypowe progowe* (ang. prototype threshold rules, PTR) [49].

4.2 Reguły typu k-NN

Reguły typu k-NNbazują na zasadzie analizy minimalnej odległości do poszczególnych prototypów, zgodnie z powszechnie znanym algorytmem klasyfikacji zwanym również k-NN[60]. W notacji regułowej, przesłance reguły odpowiada tutaj pojedynczy prototyp, natomiast konkluzji odpowiada singleton określający etykietę klasy. Najczęściej spotykanym operatorem implikacji jest operator iloczynu, który pozwala na określenie wartości wagi w określającej stopień aktywacji konkluzji. W takim przypadku uzyskiwany jest zbiór reguł postaci (4.1)

$$\begin{array}{c} \text{Jeżeli } \mathbf{x} \textit{ jest podobny do } \mathbf{p}_1 \text{ To } C(\mathbf{x}) = C(\mathbf{p}_1) \\ \text{Jeżeli } \mathbf{x} \textit{ jest podobny do } \mathbf{p}_i \text{ To } C(\mathbf{x}) = C(\mathbf{p}_i) \\ \text{Jeżeli } \mathbf{x} \textit{ jest podobny do } \mathbf{p}_{i+1} \text{ To } C(\mathbf{x}) = C(\mathbf{p}_{i+1}) \end{array} \tag{4.1}$$

gdzie ($\mathbf{x}$ *jest podobny do* $\mathbf{p}_i$) realizowane jest poprzez różne miary odległości ($D(\mathbf{x}, \mathbf{p}_i)$) lub podobieństwa ($S(\mathbf{x}, \mathbf{p}_i)$). Proces znajdowania ostatecznej odpowiedzi takiego systemu, gdy w algorytmie k-NN$k = 1$ można opisać zależnością (4.2).

$$\text{Jeżeli } \mathbf{p}' = \arg\min_i D(\mathbf{x}, \mathbf{p}_i) \text{ To } C(\mathbf{x}) = C(\mathbf{p}') \tag{4.2}$$

Wartość parametru k ma istotny wpływ na uzyskany wynik działania systemu. Odpowiada on za liczbę reguł (prototypów) branych pod uwagę podczas podejmowania decyzji, przy czym rozważanych jest k najsilniej aktywowanych prototypów. W zależności od wybranej metody agregacji konkluzji odpowiedź systemu może wówczas przyjmować różne wartości.
W przypadku klasycznego algorytmu k-NN, gdy $k > 1$ i agregacja poszczególnych konkluzji realizowana jest operatorem ($\max(\cdot)$) ostateczna odpowiedź podejmowana jest poprzez głosowanie wyznaczonych przez poszczególne najsilniej aktywowane prototypy etykiet klas. Takie rozwiązanie wymaga jednak dużej liczby prototypów co zmniejsza przejrzystość systemu, a tym samym jego przydatność jako reguł prototypowych. Dlatego też często stosowaną modyfikacją jest przekształcenie uzyskanej wartości odległości poprzez pewną funkcję $\mathcal{T}(\cdot)$ celem wyznaczenia wartości podobieństwa. Dzięki temu zwiększa się elastyczność systemu. Rozwiązanie to znane jest jako ważony lub rozmyty algorytm k-NN(ang. weighted/fuzzy/soft-k-NN) [101, 49], a poprzez modyfikację operatora $\mathcal{T}(\cdot)$ wpływa się na różne sposoby ważenia. Najczęściej w przypadku algorytmu ważonego k-NNwartość aktywacji konkluzji - wartości wag w_i przyjmuje się jako $w_i = 1/D(\mathbf{p}_i, \mathbf{x})$, $w_i = 1 - D'(\mathbf{p}_i, \mathbf{x})$ (gdzie $D'(\cdot)$ jest znormalizowaną wartością odległości).
Cechą charakterystyczną systemu k-NNjest jego równoważność z różnymi innymi systemami bazującymi na prototypach. Przykładem tego mogą być sieci RBF czy też klasyfikator Parzena, które są uzyskiwane dla $k = l$, gdzie l- to liczba wektorów zbioru treningowego oraz poprzez wykorzystanie ważonego algorytmu k-NN.
Znane są również rozwiązania klasyfikatora k-NN, w których każdemu prototypowi przypisanych jest c etykiet odpowiednio dla każdej klasy z odpowiednimi wagami $\mathbf{w}$ (jak w algorytmie klasteryzacji z powtórnym etykietowaniem, opisanym w rozdziale 6.2) [106]. Odpowiada to złożonej konkluzji, gdzie poszczególne składowe wektora

w określają stopień spełnienia danej składowej konkluzji. Jednym z alternatywnych rozwiązań jest również przypisywanie wag prototypom uwzględniające zaufanie do każdego z nich. Przykładem tego jest praca Ducha o kompetentnych komitetach [46].
Kuncheva oraz Bezdek zebrali całą rodzinę różnych metod k-NNpod wspólnym systemem zwanym *Uogólniony Klasyfikator Najbliższego Prototypu* (ang. generalized nearest prototype classifier, GNPC).GNPC składa się więc z pięciu elementów ($\mathbf{P}$,$\mathbf{W}$,$T(\cdot)$,$\mathcal{S}$,$\mathcal{A}$), które odpowiednio oznaczają:

- $\mathbf{P}$ - zbiór prototypów
- $\mathbf{W}$ - macierz wag $[l \times c]$ dla poszczególnych etykiet
- $T(\cdot)$ - monotoniczna funkcja malejąca, transformująca odległość do postaci znormalizowana miara podobieństwa np. $S(\cdot) = exp\,(-D(\cdot)^2)$ gdy $T(z) = exp(-z)$
- $\mathcal{T}$ - operator implikacji, najczęściej operator T-normy w systemach rozmytych
- $\mathcal{A}$ - operator funkcji agregującej w systemach rozmytych

Tak zbudowany system prototypowy pozwala na realizację niemal wszystkich powszechnie znanych systemów bazujących na zasadzie najbliższego sąsiada, w tym sieci LVQ, klasyfikatora Parzena, sieci RBF czy też innych algorytmów bazujących na funkcjach jądrowych (np. SVM). Co więcej, do ich realizacji wystarczą jedynie operatory w postaci $\mathcal{T} = iloczyn(\cdot)$ odpowiadający operatorowi implikacji, oraz $\mathcal{A} = max(\cdot)$ lub $suma(\cdot)$.
W systemach reguł-P pożądana jest łatwość interpretacji wyników, która sprowadza się do optymalizacji takich parametrów systemu jak:

- redukcja liczby prototypów
- redukcja wymiarowości
- selekcja odpowiedniej miary odległości lub podobieństwa
- optymalizacja położenia prototypów
- ważenie cech oraz prototypów
- dobór odpowiedniej postaci operatorów $\mathcal{S}$ oraz $\mathcal{I}$

Należy zwrócić uwagę na pewne problemy, które mogą się pojawić podczas próby optymalizacji systemów reguł prototypowych. Często bowiem liczba prototypów oraz liczba cech jest odwrotnie proporcjonalna względem dokładności klasyfikacji. Pojawia się więc problem optymalizacji wielokryterialnej, gdzie pożądany jest kompromis pomiędzy przejrzystością modelu, czyli redukcją wymiarowości i liczby prototypów, a jego dokładnością.
Innym parametrem mającym istotny wpływ na możliwości interpretacji zbudowanego modelu jest dobór operatora $\mathcal{A}()$ oraz k. Im większa jest wartość parametru k tym więcej prototypów bierze udział w podejmowaniu decyzji, a tym samym zmniejsza się łatwość zrozumienia podjętej decyzji przez system. Podobna sytuacja występuje przy doborze funkcji $\mathcal{A}()$, gdzie ostateczna decyzja jest wynikiem ważenia różnych reguł składowych. Problem ten dotyczy w szczególności reguł prototypowych typu sieci

RBF lub klasyfikatora Parzena, gdzie wszystkie prototypy jednocześnie biorą udział w podejmowaniu decyzji. Dlatego też najczęściej używanymi parametrami systemów reguł prototypowych jest $k = 1$ oraz $\mathcal{A}(\cdot) = max(\cdot)$ co pozwala na najprostszą możliwą interpretację, gdzie odpowiedź systemu podejmowana jest przez jeden prototyp.
Ogólna postać procesu strojenia (uczenia) systemu reguł prototypowych typu najbliższego sąsiada stanowi szeregowy proces przetwarzania, który może przebiegać kolejno: w pierwszym kroku realizowane jest wstępne przetwarzanie danych (np. normalizacja lub standaryzacja), następnie dokonywana jest selekcja cech, w trzecim kroku następuje wybór odpowiedniej miary odległości $D(\cdot)$ zaś w kroku czwartym dokonywana jest selekcja i optymalizacja prototypów.

4.3 Reguły prototypowe progowe

W odróżnieniu od reguł prototypowych typu k-NN, gdzie pojedynczy prototyp można było interpretować jako przesłankę reguły spełnioną w stopniu równym odległości od prototypu, tak w przypadku reguł prototypowych progowych przesłanka zdefiniowana jest przez parametry ($\mathbf{P}$,Θ,$D(\cdot)$,$\mathbf{F}$) gdzie:

- $\mathbf{P}$ - zbiór prototypów (pojedynczy prototyp odpowiada pojedynczej regule)
- Θ - wartość progu
- $D(\cdot)$ - miara odległości
- $\mathbf{f}$ - zbiór cech (lokalny lub globalny)

W związku z powyższym pojedynczą regułę można zdefiniować jako (4.3)

$$\text{Jeżeli } D(\mathbf{x}, \mathbf{p}_i) < \Theta_i \text{ To } C(\mathbf{x}) = C(\mathbf{p}_i) \tag{4.3}$$

Jej postać wskazuje na to, iż w odróżnieniu od reguł k-NNpojedynczy prototyp z progiem stanowi informatywną regułę wyznaczającą w hiperprzestrzeni wejściowej podprzestrzeń z przypisaną etykietą klasy. Pozwala to na uproszczenie interpretacji wyników oraz na zapis reguł w postaci listy typu *jeżeli ... to ...; jeżeli ... to ... jak nie to ...* lub drzewa.
Taką konstrukcję zbioru reguł można zinterpretować w sposób, gdzie każda reguła jest niezależna od pozostałych reguł. Dzięki temu możliwa staje się lokalna transformacja danych w ramach podprzestrzeni wyznaczonej przez przesłankę reguły. Między innymi pozwala to np. na lokalną selekcję cech w celu maksymalizacji możliwości uogólniania. Tej właściwości niestety nie posiadają systemy typu k-NN, które wymagają wspólnego zbioru atrybutów w całej przestrzeni wejściowej.
Jednym z pierwszych i najprostszych systemów posiadających właściwość interpretacji jako zbioru reguł jest algorytm Ograniczonej Energii Kulomba (ang. restricted columb energy, RCE) [49, 61]. Niestety jego wadą jest duży zbiór wektorów prototypowych, co znacznie utrudnia ich interpretację. Dlatego też w rozprawie omówiono oraz zaprezentowano dwa alternatywne algorytmy pozwalające na poszukiwanie odpowiedniego zbioru reguł.

4.4 Interpretacja reguł prototypowych

Podstawą działania reguł prototypowych jest analiza podobieństwa badanego przypadku do wzorców (prototypów) opisujących dane zagadnienie. Pojedynczy prototyp opisuje więc w sposób ważony fragment przestrzeni skupionej wokół prototypu, gdzie waga jest proporcjonalna do odległości pomiędzy tym prototypem, a badanym obiektem. Można więc system prototypowy interpretować w sposób, w którym pojedynczy prototyp odpowiada przesłance reguły, zaś jej konkluzją jest wartość klasy do której ten prototyp jest przypisany. Wartość podobieństwa stanowi zaś współczynnik określający pewność podjętej decyzji. Ponieważ duża część funkcji odległości może zostać poddana dekompozycji na poszczególne składowe (np. funkcje addytywne) - separowalne funkcje odległości - interpretacja działania systemu prototypowego może więc zostać przedstawiona w postaci reguł rozmytych, co omówiono w rozdziale (9).
Interpretacja systemu jako wzorców w przestrzeni wejściowej ma wiele zalet. W szczególności analiza prototypów pozwala na wstępne skoncentrowanie uwagi badawczej na wybranych wzorcach i pominięcie analizy mniej informatywnych przypadków. Przykładem tego mogą być problemy diagnozowania medycznego, gdzie analiza prototypów pozwala na wskazanie konkretnych pacjentów, a tym samym najistotniejszych i najbardziej typowych objawów chorobowych pozwalających na odróżnienie ich od osób zdrowych.
Kolejnym atutem prototypów jest ich uniwersalność w znaczeniu naturalnej interpretacji lingwistycznej. Przykładem tego może być porównanie z systemami reguł rozmytych. Lingwistyczne wnioskowanie w oparciu o systemy rozmyte zakłada przypisanie poszczególnym zmiennym lingwistycznym odpowiednich wartości lingwistycznych w postaci przymiotników przypisanych funkcją przynależności. Takie podejście czułe jest na kontekst sytuacyjny, przykładem tego może być przymiotnik *wysoki* opisujący *człowieka*, który w zależności od kontekstu - czy mamy na myśli: kobiety, mężczyzn, dzieci czy też koszykarzy przyjmuje różną reprezentację funkcyjną. Wady tej pozbawione są systemy prototypowe, w których prototyp odpowiada pewnej koncepcji, przy czym możliwa jest dekompozycja koncepcji na poszczególne składowe zmiennych lingwistycznych. Takie podejście wydaje się być dużo bardziej ogólne, co więcej - pozwala na integrację wiedzy z różnych źródeł, gdyż zawsze istnieje możliwość uszczegółowienia koncepcji poprzez dodanie jej nowych, opisujących ją atrybutów.
Wadą systemów prototypowych jest jednak redundancja informacji, która może występować dla reguł typu k-NNgdy prototypy opisane w przestrzeni wejściowej przyjmują te same wartości dla danego atrybutu. Przykładem tego mogą być prototypy opisujące osobę zdrową, dla której normalna temperatura ciała wynosi $36,6^oC$. W przypadku reguł prototypowych niemożliwe jest jednak pozbycie się tej informacji gdyż jest ona składową opisu tych koncepcji. Rozwiązaniem problemu redundancji może być segmentacja problemu i podział na podprzestrzenie w których niezależnie dokonuje się adaptacji modelu - zagadnienie to jednak nie było rozważane w ramach rozprawy. Inną możliwością jest transformacja reguł prototypowych do reguł rozmytych, gdzie w opisie przesłanki reguły możliwe jest wyłączenie części wspólnej kilku przesłanek zdekomponowanych reguł prototypowych.

4.5 Reguły prototypowe a metody bazujące na przypadkach (ang. case based reasoning)

Do metod uczenia poprzez analizę przypadków można zaliczyć: wnioskowanie pamięciowe zaproponowane przez Stanfilla i Waltza (ang. memory based reasoning, MBR) [163], uczenie w oparciu o instancje autorstwa Aha i innych (ang. instance based learning, IBL) [2] oraz wnioskowanie w oparciu o przypadki (ang. case based reasoning, CBR) [120]. Wszystkie one bazują na algorytmie k-NN, jednakże żadna z wymienionych dotychczas metod nie miała na celu reprezentacji danych w sposób zrozumiały dla człowieka.

Głównym zadaniem metod IBL jest maksymalizacja zdolności generalizacji i minimalizacji złożoności obliczeniowej algorytmu k-NN. Powoduje to, że metody należące do grupy IBL stanowią istotną grupę algorytmów mogących służyć ekstrakcji reguł-P. Niestety przeprowadzone eksperymenty [82] wskazują, iż część z metod IBL pozostawia dużo niepotrzebnych wektorów w procesie uczenia, koncentrując się nie na uzyskaniu przejrzystości modelu, lecz jedynie na maksymalizacji jego zdolności uogólniania.

CBR powszechnie stosowany jest do rozwiązywania różnych problemów analizy przypadków, gdzie pojedyncze sytuacje odpowiadają wystąpieniu określonych zjawisk. Innymi słowy CBR bazuje na rozwiązywaniu nowych problemów na podstawie podobieństwa do już rozwiązanych zagadnień zgromadzonych w bazie wiedzy. CBR między innymi używany jest w analizie przepisów prawa precedensowego, wiele wdrożeń można również zaobserwować w medycynie np. [148]. W systemach CBR główny nacisk położony jest również na maksymalizację dokładności predykcji, jednakże poszczególne przypadki traktowane są jako pojedyncze reguły postępowania, a baza wiedzy jest często inkrementacyjnie rozbudowywana. Związane jest to z procesem eksploracji danych online, gdzie często każdorazowy sukces lub porażka predykcji zakończony jest umieszczeniem tego przypadku w bazie wiedzy z odpowiednio zmodyfikowaną etykietą. Wykorzystanie algorytmu k-NNw CBR związane jest z koniecznością analizy podjętej decyzji, a jak opisano w poprzednim podrozdziale metoda k-NNpozwala na jej łatwą interpretację.

Główną różnicą modeli CBR w stosunku do reguł-P jest brak przejrzystości systemów CBR, co związane jest z dużą liczbą przypadków zgromadzonych w bazie wiedzy. Problem ten powoduje również zmniejszenie szybkości działania oraz duże zapotrzebowanie na zasoby tego typu systemu. Dlatego powszechnie stosuje się różne metody mające na celu przyspieszenie procesu znajdywania najbliższych sąsiadów. Do typowych rozwiązań należą tutaj algorytm Kd-tree [176] lub inne metody drzew decyzji jak C4.5, które wykorzystywane są do wstępnej indeksacji i selekcji przypadków [139].

Rozdział 5

Miary odległości

Miary odległości, jak i miary podobieństwa stanowią bazę większości metod inteligencji obliczeniowej. W literaturze opisywana jest duża liczba możliwych miar odległości, jednakże praktyka metod sztucznej inteligencji pokazuje, że nie wszystkie z nich muszą spełniać własności metryki opisywane warunkami (5.1).
W prezentowanej rozprawie terminy takie, jak podobieństwo lub odległość są używane w szerszym znaczeniu niż matematycznym i muszą jedynie spełniać warunek *izolacji* dla odległości. Założenie takie podyktowane jest zarówno analogią do procesów kognitywnych zachodzących w naszych mózgach, gdzie przykładowo „niebo" kojarzy się z kolorem niebieskim, ale odwrotna relacja nie jest już tak silna $D\left(niebo, niebieski\right) \neq D\left(niebieski, niebo\right)$ (gdzie $D(\cdot)$ oznacza funkcję odległości) nie jest więc spełniony warunek symetrii; podobnie mówiąc, że dziecko jest podobne do ojca oraz podobne do matki nie oznacza faktu, że matka i ojciec są do siebie podobni; oraz z praktyki metod inteligencji obliczeniowej, gdzie, jak pokazano w rozdziale 9 transformacja iloczynu trójkątnej lub trapezoidalnej funkcja przynależności systemów rozmytych do postaci funkcji odległości nie spełniają warunku nierówności trójkąta.

$$\begin{aligned} &D\left(\mathbf{x}, \mathbf{p}\right) \geq 0, \text{ równość, wtedy i tylko wtedy gdu } \mathbf{x} = \mathbf{p} \text{ (izolacja)} \\ &D\left(\mathbf{x}, \mathbf{p}\right) = D\left(\mathbf{p}, \mathbf{x}\right) \text{(symetria)} \\ &D\left(\mathbf{x}, \mathbf{p}\right) \leq D\left(\mathbf{x}, \mathbf{y}\right) + D\left(\mathbf{y}, \mathbf{p}\right) \text{(nierówność trójkąta)} \end{aligned} \tag{5.1}$$

Ogólna taksonomia różnych neuronowych funkcji transferu zawierająca funkcje odległości i podobieństwa może być znaleziona w [50, 124]. Jednakże nie wyczerpuje ona wszystkich znanych miar, o pewnych zastosowaniach specyficznych miar odległości, jak w problemach wizji komputerowej (ang. computer vision) donoszono również w pracach [41, 78, 129]. W rozprawie przedstawiono jedynie najpopularniejsze miary z ich własnościami, które jak dotąd zostały zastosowane w systemach reguł progowych z bardzo dobrymi rezultatami.
W swojej pracy Duch i Jankowski [50] zaproponowali różne metody podziału miar odległości, jednakże najbardziej istotną metodą z punktu widzenia rozprawy wydaje się być podział ze względu na wpływ danych na wartość miary odległości.

5.1 Deterministyczne miary odległości

Deterministyczne funkcje odległości charakteryzują się brakiem wpływu rozkładu danych w przestrzeni na wartość funkcji odległości. Innymi słowy rozkład danych treningowych nie wpływa na wartość odległości mierzonej pomiędzy parą wektorów.

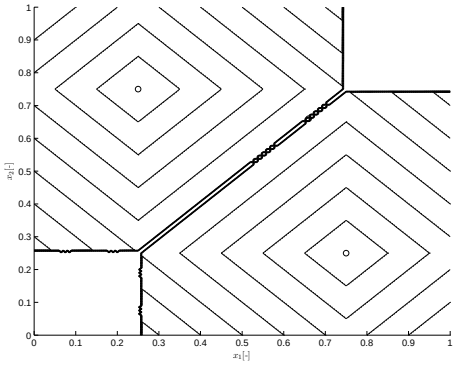
Najpopularniejszym przykładem takich funkcji jest metryka Euklidesa (norma L_2) (5.2)

$$D(\mathbf{x}, \mathbf{p}) = \sqrt{\sum_{i=1}^{n} (x_i - p_i)^2} \tag{5.2}$$

gdzie $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ i $\mathbf{p} = [p_1, p_2, ..., p_n]^T$ są wektorami pomiędzy którymi odległość D jest liczona. Metryka ta ma istotną właściwość z punktu widzenia reguł prototypowych, a mianowicie stosując regułę najbliższego sąsiada uzyskiwana granica decyzji pomiędzy dwoma prototypami jest zawsze hiperpłaszczyzną. Dzięki temu znacznie upraszcza się proces wnioskowania i zrozumienia rezultatów podjętych decyzji. Innym przykładem metryki deterministycznej jest odległość Manhattan, znana również jako norma L_1 gzie odległość jest obliczana jako suma różnic projekcji wektorów na osie układu współrzędnych (5.3).

$$D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} |x_i - p_i| \tag{5.3}$$

Ważną negatywną cechą normy L_1 w zastosowaniu do algorytmu k-NNjest możliwość wystąpienia impasów. Właściwość ta wynika z faktu, iż w odróżnieniu od innych miar odległości obszar równej odległości pomiędzy dwoma prototypami nie jest jak w przypadku normy L_2 hiperpłaszczyzną lub ogólnie figurą płaską, lecz granica ta w pewnych obszarach przyjmuje postać podprzestrzeni o równej odległości od pary prototypów, co uniemożliwia jednoznaczne podjęcie decyzji. Przedstawia to rys.(5.1)



Rysunek 5.1: Przykład wykorzystania odległości Manhattan oraz impasu powstałego po jej wykorzystaniu (obszar nie zakreślony)

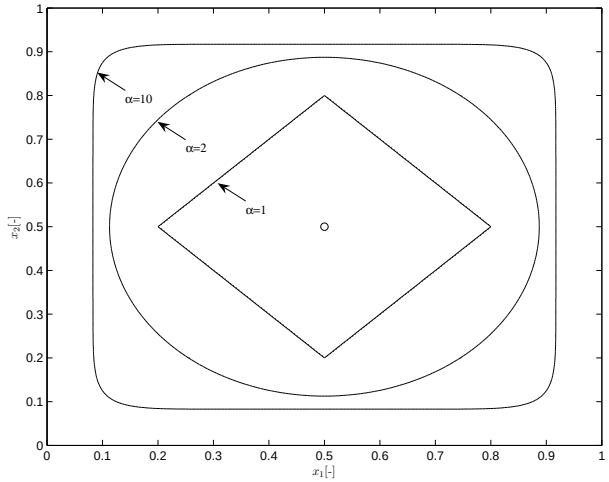
Norma L_∞ znana również jako odległość Czebyszewa opisywana jest równaniem (5.4)

$$D(\mathbf{x}, \mathbf{p}) = \max_{i=1..n} |x_i - p_i| \tag{5.4}$$

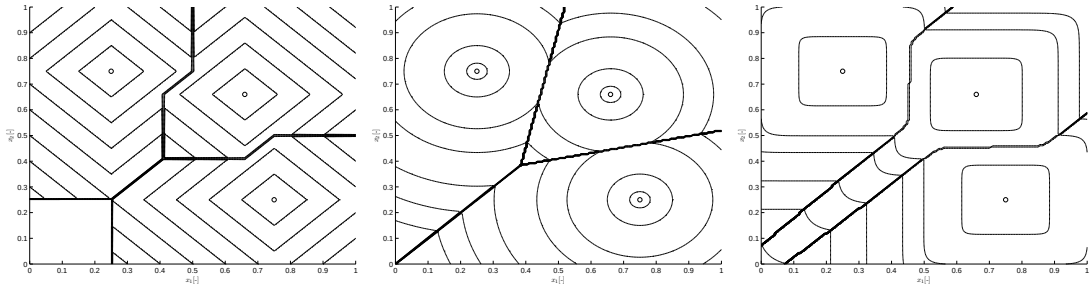
Funkcja ta, z punktu widzenia reguł prototypowych ma istotne znaczenie, pozwalając na uzyskiwanie klasycznych reguł ostrych w zapisie prototypowym.
Przedstawione dotychczas miary odległości stanowią poszczególne przypadki ogólnej metryki Minkowskiego opisanej równaniem (5.5), gdzie zmiana parametru α wpływa na kształt konturu stałej odległości od prototypu. (5.2)

$$D(\mathbf{x}, \mathbf{p}) = \left(\sum_{i=1}^{n} |x_i - p_i|^\alpha \right)^{1/\alpha} \tag{5.5}$$

Przez odpowiednie określenie wartości parametru α możliwe jest uzyskanie odległości Manhatan dla $\alpha = 1$, podobnie dla $\alpha = 2$ otrzymuje się odległość Euklidesa, natomiast dalszy wzrost wartości w granicy gdy $\alpha \to \infty$ prowadzi do odległości Czebyszewa. Interpretacja rys.(5.2) zmierza do wniosku, że wzrost wartości α odpowiada zwiększeniu wypukłości funkcji odległości, tym samym wzrostowi pola recepcyjnego pojedynczego prototypu. Powoduje to, że zmianie ulega kształt granicy decyzji klasyfikatora 1NN, co obrazuje rys.(5.3). W praktyce analizy danych bardzo często wartość α poddawana jest adaptacji w procesie uczenia jako jeden z parametrów modelu, tak by zmaksymalizować dokładność uzyskanych wyników.



Rysunek 5.2: Kształ odległości Minkowskiego dla trzech różnych wartości parametru α: 1,2,10



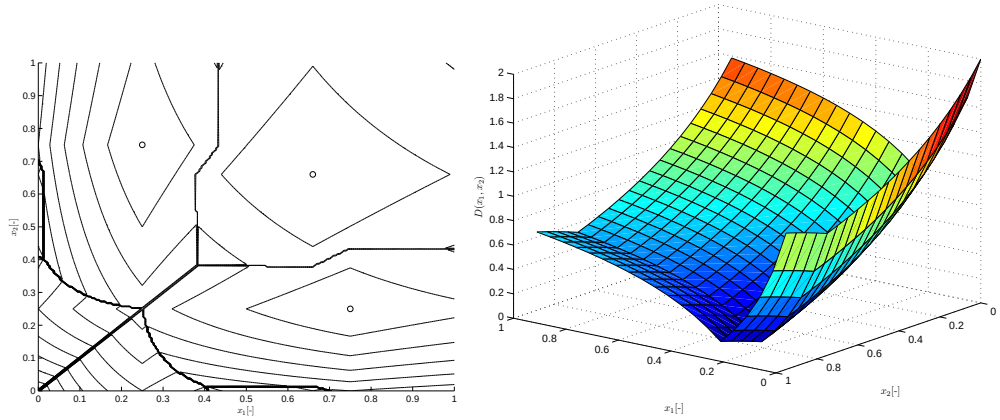
Rysunek 5.3: Kształt granicy decyzji dla klasyfikatora 1NN dla trzech różnych wartości α odległości Minkowskiego: 1,2,10

Pewną popularną miarą odległości, dającą dla niektórych zbiorów danych bardzo dobre wyniki, jest miara Canberra (5.6) należąca do grupy miar korelacyjnych rys.(5.4).

$$D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} \frac{|x_i - p_i|}{|x_i| + |p_i|} \tag{5.6}$$

Miara ta ma pewne specyficzne cechy, jest niesymetryczna (nie jest więc miarą metryczną) oraz dla wektora referencyjnego położonego w początku układu współrzędnych przyjmuje stałą wartość równą 1. Inną cechą tej miary jest możliwość

występowania symbolu nieoznaczonego jako wartość odległości w przypadku wystąpienia dzielenia przez 0. Cechy te powodują że praktyczne zastosowania w/w funkcji choć często dające dobre rezultaty powinno być uważnie rozpatrzone. Inną, popularną

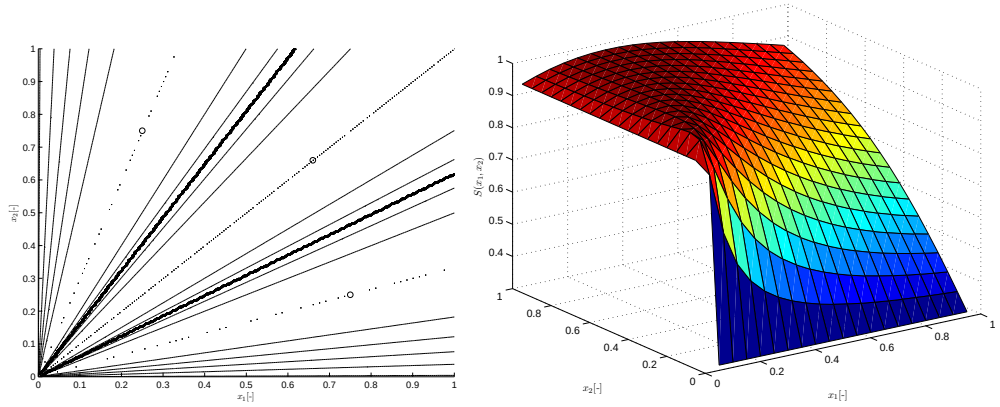


Rysunek 5.4: Kształt granicy decyzji oraz wykres powierzchni odległości Canbera

miarą powszechnie stosowaną do analizy tekstów jest odległość cosinusowa, będąca w rzeczywistości miarą podobieństwa zdefiniowaną jako (5.7).

$$D\left(\mathbf{x},\mathbf{p}\right)=\frac{\mathbf{x}^{T}\mathbf{p}}{\left|\mathbf{x}\right|\left|\mathbf{p}\right|} \tag{5.7}$$

Funkcja ta mierzy cosinus konta pomiędzy dwoma wektorami $\mathbf{x}$ oraz $\mathbf{p}$ jako ich iloczyn skalarny znormalizowany przez iloczyn długości tych wektorów. Własność ta jest używana do porównywania dokumentów, gdyż przedstawia ona odległość jako relatywny rozkład jej elementów składowych, uniezależniając wartość od długości tekstów. Kształt granic decyzji oraz kształt powierzchni odległości cosinusowej przedstawia rys.(5.5) Podobnie do analizy tekstów używa się miar znormalizowanych, uniezależniając wartość



Rysunek 5.5: Granica decyzji oraz kształt cosinusowej funkcji odległości

odległości od długości wektorów, co zapisuje się jako (5.8).

$$\overline{D}(\mathbf{x},\mathbf{p})=D\left(\mathbf{x}/_{|\mathbf{x}|},\mathbf{p}/_{|\mathbf{p}|}\right) \tag{5.8}$$

5.2 Miary zależne od rozkładu danych

Miary zależne od rozkładu danych mają tą właściwość, że na wartości odległości przez nie przyjmowane ma wpływ wzajemne położenie innych wektorów, stanowiących zbiór przypadków do których należą wektory pomiędzy którymi liczona jest ta odległość. W grupie tej można wyznaczyć trzy podgrupy: ważone funkcje odległości, probabilistyczne miary odległości oraz odległości heterogeniczne.
Problematyka adaptacji miar odległości do badanego zagadnienia jest coraz szerzej dostrzegana przez różne środowiska naukowe. W szczególności dotyczy to problemów analizy tekstów [109], obróbki obrazu [27, 129] czy też zastosowań biomedycznych [188]. Wynikiem tego są różnego rodzaju warsztaty dotyczące problematyki optymalizacji miar odległości odbywające się podczas konferencji jak np. „Learning to Compare Examples" odbywająca się w ramach konferencji NIPS'06.

5.2.1 Ważone miary odległości

Do tej rodziny miar należą odległości przedstawione w poprzednim rozdziale, z tą jednak różnicą, że wpływ poszczególnych atrybutów jest ważony poprzez czynnik w_i.

$$D\left(\mathbf{x},\mathbf{p};\mathbf{w}\right)=\left(\sum_{i=1}^{n} w_i\left|x_i-p_i\right|^{\alpha}\right)^{1/\alpha} \tag{5.9}$$

Wektor $\mathbf{w}$ jest uzyskiwany z rozkładu danych i ma wartości niezależne dla każdej z cech. Problem wyznaczania wektora $\mathbf{w}$ poruszali w swoich pracach m.in. Jankowski, Martinez oraz Dudani [124, 182, 62], natomiast ogólne porównanie różnych metod ważenia cech dla klasyfikatora k-NNprzedstawił Wettschereck i inni w [177].
W literaturze spotkać można dwa podejścia związane z ważeniem cech - globalne oraz lokalne. W pierwszym podejściu wyznaczane są wspólne wagi dla wszystkich prototypów, osobno dla każdej z cech. Podejście to ma tą cechę, iż podaje ogólny poziom istotności cechy w procesie dyskryminacji dla analizowanego zbioru. Jest to więc w pewnych problemach bardzo pożądana wiedza. Możliwa jest też selekcja lub ranking cech w oparciu o tak wyznaczone współczynniki. Druga możliwość - ważenie lokalne wyznacza odpowiednie wagi dla każdej z cech i dla każdego z prototypów osobno, dzięki czemu możliwe jest uzyskanie większej elastyczności modelu, a uzyskane prototypy lepiej oddają lokalne rozkłady danych. Wadą podejścia lokalnego jest wzrost liczby stopni swobody budowanego modelu $(l\cdot n)$ w stosunku do $(l+n)$ dla modelu z globalnym ważeniem cech. Ma to szczególnie istotne znaczenie przy analizie pewnej klasy problemów zawierających małą liczbę wektorów treningowych (m).
Przedstawione funkcje odległości mają jedną istotną wadę - używając ich zakłada się bowiem niezależność cech. Założenie to jest uzasadnione, jeżeli dążymy do uzyskania dużej łatwości w interpretacji modelu, podobnie jak w rozmytych lub neuronowo-rozmytych systemach regułowych, gdzie niezależność poszczególnych składowych pozwala na wykorzystanie wnioskowania opartego o prawa logiki rozmytej. Wadą takiego podejścia jest niedokładne odwzorowanie rozkładu danych w przestrzeni, w szczególności jeżeli występuje korelacja pomiędzy poszczególnymi cechami. Dlatego też często stosowanym rozwiązaniem są macierze wag $\mathbf{W}$, których elementy $\mathbf{w}_{ij}$ definiują relacje pomiędzy cechami i-tą i j-tą. Jeżeli macierz $\mathbf{W}$ określona jest jako macierz kowariancji $\mathbf{W}=\mathbf{A}$, wówczas odległość (5.10) nazywana jest odległością Mahalanobisa. Przy czym $\mathbf{A}$ jest macierzą symetryczną względem diagonalnej oraz dodatnio określoną.

$$D(\mathbf{x}, \mathbf{p}; \mathbf{W}) = (\mathbf{x} - \mathbf{p})^T \mathbf{W} (\mathbf{x} - \mathbf{p}) \tag{5.10}$$

Jak już wspomniano wartości macierzy **A** uzyskiwane są poprzez analizę korelacji danych, zarówno globalną jak i lokalną. Powszechne są również podejścia wykorzystujące różne metody optymalizacji jak gradientową, stosowaną powszechnie w sieciach RBF [151] lub bazująca na algorytmie EM (ang. expectation maximization). Przykładem zastosowania macierzy **W** w algorytmie k-NNjest metoda *Large Margin k-NN* zaproponowana przez Weinberger, Blitzer i Saulin w [174].
Przedstawione dotychczas funkcje odległości przeznaczone są dla atrybutów ciągłych, jednakże analiza realnych zbiorów danych często sprowadza się do problemu cech dyskretnych, symbolicznych lub binarnych. W takich przypadkach powszechnie stosowanym rozwiązaniem jest odległość Hamminga opisana równaniem (5.11)

$$D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} \delta(x_i, p_i) \tag{5.11}$$

gdzie $\delta(\cdot)$ jest deltą Kroneckera

5.2.2 Probabilistyczne miary odległości

Szczególny problem dla wszystkich metod bazujących na algorytmie k-NNstanowiły cechy o nieciągłej dziedzinie, w szczególności cechy symboliczne, w których typowe funkcje odległości nie zdają egzaminu. Pewnym rozwiązaniem tego problemu były funkcje odległości z rodziny funkcji Hamminga, jednakże uzyskiwane rezultaty często odbiegały od oczekiwań. Ich alternatywą stały się probabilistyczne miary odległości, wywodzące się z analizy możliwości generalizacji algorytmu k-NN. Jedną z pierwszych probabilistycznych funkcji odległości była miara, którą zaproponował Short i Fukunaga (miara SFN) [157] opisana równaniem (5.12)

$$D(\mathbf{x}, \mathbf{p}) = \sum_{k=1}^{c} p(C_k|\mathbf{p}) \left| p(C_k|\mathbf{p}) - p(C_k|\mathbf{x}) \right| \tag{5.12}$$

SFM została wyznaczona jako miara lokalna, która bazowała na minimalizacji wartości oczekiwanej różnicy pomiędzy błędem klasyfikacji algorytmu k-NNze skończoną i nieskończoną liczbą próbek (wartość teoretyczna) - innymi słowy bazowała na minimalizacji asymptotycznego i skończonego ryzyka.
Inną probabilistyczną miarą odległości wyprowadzoną przy założeniu bezpośredniej minimalizacji błędu klasyfikacji była miara Minimum Risk Metric (MRM) (5.13) zaproponowana przez Blanzieri i Ricci w [20, 21].

$$D(\mathbf{x}, \mathbf{p}) = \sum_{k=1}^{c} p(C_k|\mathbf{p}) \left| 1 - p(C_k|\mathbf{x}) \right| \tag{5.13}$$

Stanfill i Waltz w [162] zaproponowali odległość Value Difference Metric (VDM). Ich propozycja nie wywodziła się z analizy teoretycznej, jednakże przeprowadzone eksperymenty wykazały jej bardzo dużą skuteczność. Oryginalna miara VDM była zaproponowana bezpośrednio dla danych symbolicznych jako (5.14).

$$D(X, P) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{c} \left(\frac{N(x_i, c_j)}{N(x_i)} \right)^2} \sum_{j=1}^{c} \left(\frac{N(x_i, c_j)}{N(x_i)} - \frac{N(p_i, c_j)}{N(p_i)} \right)^2 \tag{5.14}$$

gdzie $N(x_i)$ oznacza liczbę wystąpień wartości x dla cechy i (x_i) w zbiorze treningowym oraz $N(x_i, c_j)$ oznacza liczbę wystąpień wartości x_i dla i-tej cechy j-tej klasy c_j, a ostatecznie $N(x_i, c_j)/N(x_i)$ stanowi ocenę prawdopodobieństwa aposteriori $p(c|x)$. Wówczas odległość VDM może zostać również zapisana jako (5.15)

$$D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{c} p(c_j|x_i)^2} \sum_{j=1}^{c} (p(c_j|x_i) - p(c_j|p_i))^2 \tag{5.15}$$

Oryginalna postać odległości VDM została następnie zmodyfikowana przez Cost i Salzberga w [32] do postaci metryki MVDM (ang. modified value difference metric) (5.16), gdzie pominięto pierwszy czynnik równania odległości.

$$D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{c} (p(c_j|x_i) - p(c_j|p_i))^{\alpha} \tag{5.16}$$

gdzie α oznacza wartość dobieraną empirycznie dla konkretnego problemu.

5.2.3 Heterogeniczne miary odległości

Poważnym zagadnieniem problemów klasyfikacyjnych są zbiory o niejednorodnych typach atrybutów, w których część cech jest ciągła, a część symboliczna lub dyskretna. Wówczas żadna z dotychczas prezentowanych funkcji odległości nie pozwala na uzyskiwanie pożądanych wyników. Problem ten studiowali Wilson i Martinez w [181] i zaproponowali szereg miar heterogenicznych, które stosują różne typy odległości dla różnych typów cech. Ogólna idea zaproponowana przez autorów wykorzystuje cechę addytywności funkcji odległości, gdyż większość spośród przedstawionych funkcji uwzględniając miary VDM, Hamminaga czy Minkowskiego składają poszczególne komponenty cech poprzez ich sumowanie.

$$\begin{aligned} D^2(\mathbf{x}, \mathbf{p}) = D^2_{Mink}(\mathbf{x_a}, \mathbf{p_a}; \mathbf{w_a}) + D^2_{VDM}(\mathbf{x_b}, \mathbf{p_b}; \mathbf{w_b}) \\ + D^2_{Hamm}(\mathbf{x_c}, \mathbf{p_c}; \mathbf{w_c}) \end{aligned} \tag{5.17}$$

gdzie $\mathbf{a}, \mathbf{b}, \mathbf{c}$ są wektorami o indeksach cech odpowiednio ciągłych, symbolicznych i nominalnych, a $\mathbf{w}$ jest wektorem normalizującym (wektorem wag) dla poszczególnych cech. Przykładem takich odległości heterogenicznych są *Heterogeneous Euclidean-Overlap Metric* (HEOM), gdzie połączono odległość Euklidesa z odległością Hamminga oraz *Heterogeneous Value Difference Metric* (HVDM) [181], w której sumowane są odległości Euklidesa z odległością VDM. W obydwu tych przypadkach uwzględniono również problem brakujących wartości poprzez dodanie 1 dla atrybutów z brakującymi wartościami.

Innym rozwiązaniem zaproponowanym przez Wilsona i Martineza była adaptacja odległości probabilistycznej do analizy atrybutów ciągłych, bazując na wyrażeniu (5.16). Zasugerowali oni trzy metody nieparametrycznej estymacji prawdopodobieństwa wykorzystującej: dyskretyzację (naiwna dyskretyzacja stałej szerokości) - odległość DVDM, interpolację wartości uzyskanych z miary DVDM - odległość IVDM oraz estymację bazującą na okienkowaniu - zwaną odległością WVDM. Dzięki takiemu rozwiązaniu zniknął problem normalizacji cech występujący w przypadku wykorzystania cechy addytywności odległości jak w (5.17), gdyż wszystkie analizowane atrybuty transformowane są wstępnie do wspólnej przestrzeni prawdopodobieństwa.

5.3 Nowe probabilistyczne miary odległości

Wykorzystania odległości probabilistycznych w zastosowaniu do metod reguł prototypowych analizowane były również podczas realizowanej rozprawy. Autor w pracach [180, 16, 178] zaproponował rozszerzenie istniejących miar heterogenicznych o dodatkową grupę bazującą na nieparametrycznych metodach estymacji prawdopodobieństwa. Grupa zaproponowanych miar dotyczyła wykorzystania odległości prawdopodobieństwa dla atrybutów ciągłych. W rezultacie badań zaproponowano trzy miary (GVDM, LVDM oraz PVDM) wykorzystujące do estymacji różne metody wywodzące się z metody okien Parzena [173] i przeprowadzono analizę przydatności zaproponowanych metryk.

5.3.1 Nieparametryczne metody estymacji prawdopodobieństwa

Do realizacji probabilistycznych funkcji odległości niezbędna jest estymacja prawdopodobieństwa dla każdego z atrybutów. W tym celu zweryfikowano trzy różne metody: wygładzania Gausowskiego, prostokątnych okien Parzena oraz okien przesuwnych, w wyniku czego uzyskano trzy nowe miary odpowiednio GVDM, LVDM oraz PVDM.

Wygładzanie Gaussowskie. Miara GVDM, gdzie dla każdego z atrybutów f, wartość prawdopodobieństwa $p(c_j|x_{kf})$ dla wektora x_k wyznaczana jest wg. zależności (5.18)

$$\begin{gathered} p(c_j|x_{kf}) = \beta \cdot \sum_{i=1}^{m_j} \exp\left(-\gamma(x_{if} - x_{kf})^2\right) \\ \beta = 1/\sum_{a=1}^{C}\left(\sum_{i=1}^{m_j} \exp\left(-\gamma x_{if}^2\right)\right) \end{gathered} \tag{5.18}$$

gdzie β jest parametrem normalizującym zapewniającym by $\sum_j p(c_j|x_f) = 1$, x_{kf} to cecha f k-tego wektora, m_j to zbiór wektorów zbioru treningowego należących do klasy j-tej, natomiast γ jest parametrem definiowanym przez użytkownika określającym szerokość funkcji Gaussa.

Prostokątne okna Parzena. Miara LVDM w której $p(c_j|x_kf)$ wyznacza się stosując lokalną estymację na zasadzie częstości według zależności (5.19)

$$p(c_j|x_{kf}) = \frac{N(w^{x_{kf}}, c_j)}{N(w^{x_{kf}})} \tag{5.19}$$

gdzie $N(w^{x_{kf}}, c_j)$ jest liczebnością wektorów z klasy c_j znajdujących się w obszarze określonym parametrami $w^{x_{kf}} = [a-\sigma/2, a+\sigma/2]$ i $N(w^{x_{kf}})$ jest odpowiednio całkowitą liczbą wektorów w tym obszarze, a σ to parametr określający szerokość okna.

Okna przesuwne. Miara PVDM - gdzie do estymacji wartości prawdopodobieństwa $p(c_j|x_{kf})$ budowany jest histogram o nakładających się przedziałach, a wartość prawdopodobieństwa wyznaczana jest jako średnia wszystkich przedziałów, które obejmują badany wektor (5.20).

$$p(c_j|x_{kf}) = \frac{1}{z}\sum_{i=1}^{z} \frac{N^i(x_{fk}, c_j)}{N^i(x_{fk})} \tag{5.20}$$

gdzie z jest liczbą wszystkich przedziałów zawierających badany wektor x_{kf}, natomiast $N^i(x_{fk}, c_j)$ oraz $N^i(x_{fk})$ to odpowiednio wartości prawdopodobieństwa znajdujące się w i tym przedziale.

5.3.2 Wykorzystanie probabilistycznych miar odległości do systemów reguł prototypowych

Odległość heterogeniczne z rodziny VDM mają pewną istotną cechę - możliwe jest bowiem pokazanie, iż w rzeczywistości miary te polegają na mapowaniu wektorów danych dla każdej z cech niezależnie na $c-1$ wymiarową przestrzeń prawdopodobieństwa, gdzie następnie stosowana jest metryka Minkowskiego (5.21)

$$\begin{gathered} D(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{c} (p(c_j|x_i) - p(c_j|p_i))^q \\ \Downarrow \\ D(\mathbf{x}, \mathbf{p}) = \sum_{k=1}^{c \cdot n} (g(x_k) - g(p_k))^q \end{gathered} \tag{5.21}$$

gdzie $g(\cdot)$ jest funkcją realizującą odwzorowanie zapisane zależnością

$$g(x_k) = \begin{cases} p(c_{1\ldots c}|x_1) \\ p(c_{1\ldots c}|x_2) \\ p(c_{1\ldots c}|x_n) \end{cases} \tag{5.22}$$

Jednak ze względu na liniową zależność atrybutów pochodzących z odwzorowania $p(c_{1\ldots c}|x_1)$ gdyż $\sum_{j=1}^{c} p(c_j|x_1) = 1$ wystarczające jest odwzorowanie pojedynczego atrybutu w postaci $c-1$ nowych atrybutów [47].

5.3.3 Porównanie heterogenicznych miar odległości w zastosowaniu do reguł prototypowych

W celu sprawdzenia i porównania różnych miar odległości heterogenicznych dokonano weryfikacji opisanych miar na realnych zbiorach danych. Testów porównawczych dokonano stosując 10-cio krotną walidację krzyżową, dla różnych zestawów parametrów metod estymacji i algorytmu klasyfikacji CFCM+LVQ opisanego w rozdziale (6.3). Wyniki porównania przedstawione w tabeli (5.1), zawierają najlepsze rezultaty uzyskane przez każdą z miar odległości oraz liczbę wybranych prototypów (l). Zbiory wykorzystane do testowania opisano w rozdziale (12).

Uzyskane rezultaty wskazują, iż zaproponowane miary heterogeniczne zwiększają dokładność klasyfikacji, co w znacznym stopniu uwydatnia się dla zbioru *jonosfera* oraz *sonar*. Ponadto dobór odpowiedniej metryki pozwala na redukcję liczby prototypów przy jednoczesnym wzroście dokładności klasyfikacji. Godnym zwrócenia uwagi jest fakt, iż miara PVDM wymagała doboru dwóch parametrów jednocześnie co znacznie zwiększyło złożoność obliczeniową tej metody, przy porównywalnych wynikach uzyskanych miarami GVDM oraz LVDM.

Przeprowadzona analiza wskazuje na dużą skuteczność zaproponowanych metryk. Ich wykorzystanie nie ogranicza się jedynie dla grupy metod reguł prototypowych, pozwalając na jej użycie również wśród innych metod inteligencji obliczeniowej, w szczególności gdy nadrzędnym celem jest maksymalizacja dokładności klasyfikacji.

	HVDM		GVDM		LVDM		PVDM	
	Dokładność	l	Dokładność	l	Dokładność	l	Dokładność	l
Choroby serca	83.12 ± 5.85	2	83.83 ± 5.91	2	84.49 ± 5.86	2	84.85 ± 5.99	3
Jonosfera	86.37 ± 6.62	7	91.21 ± 5.03	2	93.21 ± 4.62	2	91.82 ± 5.32	3
Cukrzyca	74.74 ± 5.23	2	74.88 ± 5.62	2	75.52 ± 3.97	2	75.79 ± 4.24	3
Rak piersi	97.66 ± 1.83	3	97.81 ± 1.84	3	97.81 ± 1.83	3	97.81 ± 1.83	3
Wyrostek robaczkowy	86.61 ± 13.32	2	85.79 ± 11.39	2	86.79 ± 10.23	2	86.79 ± 10.23	2
Sonar	60.02 ± 21.87	2	64.79 ± 19.61	2	70.40 ± 15.24	5	68.26 ± 18.44	2
Choroby wątroby	59.42 ± 6.49	6	59.98 ± 5.40	2	59.93 ± 7.71	2	65.52 ± 8.33	2
Irysy	96.00 ± 6.44	6	95.33 ± 5.49	3	96.67 ± 4.71	3	96.00 ± 4.66	3
Winorośl	96.08 ± 3.77	5	98.33 ± 2.68	5	96.73 ± 4.58	4	98.89 ± 2.34	7

Tablica 5.1: Porównanie jakości klasyfikacji systemów reguł prototypowych typu k-NNz wykorzystaniem różnych probabilistycznych miar odległości.

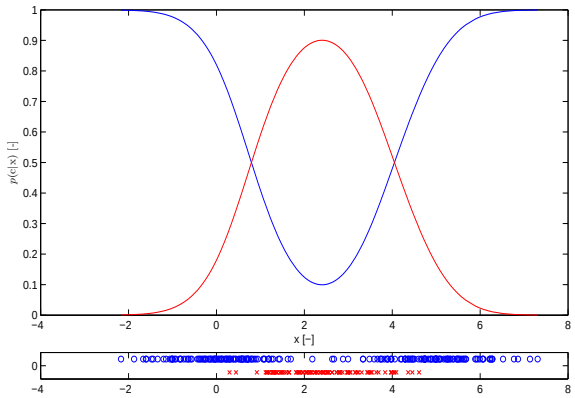
5.4 Wnioski

Analiza literaturowa wskazuje, iż dobór odpowiedniej metryki pozwala w znacznym stopniu na maksymalizację dokładności klasyfikacji oraz generalizacji budowanego modelu. Proces ten może być realizowany na podstawie wiedzy a'priori bądź też poprzez metody metauczenia, które pozwalają na jego automatyzację.
W praktyce stosowania miar heterogenicznych najczęściej spotykanym rozwiązaniem jest wykorzystanie metody omówionej w podrozdziale 5.3.2, gdzie następuje odwzorowanie pojedynczego atrybutu w postaci $c-1$ cech. Należy tutaj zwrócić uwagę na pewne teoretyczne własności dwóch występujących tutaj podejść:

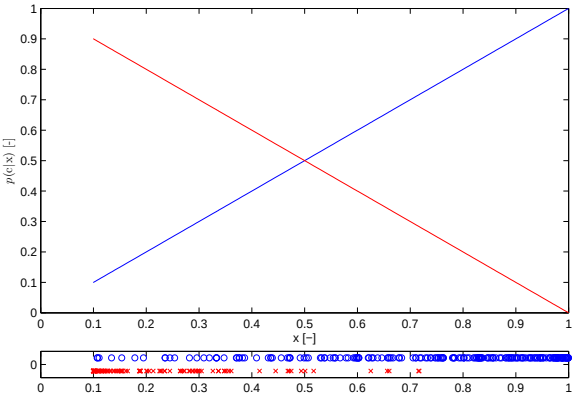
1. mapowanie do postaci prawdopodobieństwa jedynie cech symbolicznych i nominalnych (HEOM, HVDM)

2. mapowanie wszystkich cech do przestrzeni prawdopodobieństwa (miary DVDM, IVDM, GVDM, PVDM, LVDM)

Pierwsze podejście najczęściej wykorzystuje metrykę Minkowskiego w nowej rozszerzonej przestrzeni. Pojawia się jednak problem normalizacji lub ważenia atrybutów ciągłych nie poddanych transformacji, co powoduje niejednorodność różnych cech. W podejściu drugim omówiony problem nie występuje, gdyż wszystkie cechy znajdują się we wspólnej przestrzeni prawdopodobieństwa, co pozwala na pominięcie problemu normalizacji. Jednakże wadą podejścia nr 2 jest konieczność wyznaczenia optymalnych parametrów estymacji, co jest równie istotnym problemem jak wyznaczanie wag. Obydwie te metody wydają się być więc równoważne.
Pewną korzyścią wynikającą z transformacji wartości atrybutów ciągłych do postaci prawdopodobieństwa w przypadku systemów reguł prototypowych jest możliwość redukcji liczby prototypów. Przykładowo, sytuacja taka może zajść, jeżeli rozkład wartości atrybutu f przyjmuje postać jak na rys.(5.6) wówczas do poprawnej klasyfikacji takiego zagadnienia niezbędne są trzy prototypy. Po transformacji do postaci prawdopodobieństwa nowa cecha przyjmuje postać jak na rys.(5.7). Dzięki temu możliwa jest redukcja złożoności modelu z trzech do dwóch prototypów przy zachowaniu podobnej dokładności klasyfikacji. Transformacja do wspólnej przestrzeni prawdopodobieństwa ma jednak pewne wady. Mianowicie odwzorowanie odwrotne - położenia prototypów z wartości prawdopodobieństwa do wartości wejściowej jest niejednoznaczne, gdyż możliwe jest znalezienie wiele różnych wektorów o równej wartości funkcji prawdopodobieństwa. Problem ten szczególnie dotyczy atrybutów ciągłych. Dlatego też spojrzenie na omówiony problem z perspektywy łatwości



Rysunek 5.6: Rozkład prawdopodobieństwa oraz wykres wartości atrybutu f dla problemu klasyfikacji dwóch klas



Rysunek 5.7: Rozkład prawdopodobieństwa atrybutu f po transformacji z wykorzystaniem odległości probabilistycznych

interpretacji uzyskanych wyników prowadzi do wniosku, iż w praktyce wykorzystanie odległości heterogenicznych typu HEOM lub HVDM ułatwia analizę uzyskanych wyników. Natomiast zaproponowane miary realizujące mapowanie atrybutów ciągłych (IVDM,DVDM,GVDM,LVDM,PVDM) powinny być używane bądź ze względu na dokładność wyników (możliwa poprawa dokładności klasyfikacji), bądź też gdy informacje o rozkładzie prawdopodobieństwa atrybutów są wystarczającym źródłem informacji, pozwalającym na interpretację wyników.

Rozdział 6

Selekcja i optymalizacja prototypów dla reguł najbliższego sąsiada

Problem selekcji i optymalizacji prototypów dla algorytmu k-NNjest niezmiernie istotny, gdyż bezpośrednio wpływa na uzyskaną zdolność uogólniania algorytmu, jednocześnie zmniejszając jego złożoność obliczeniową. Problem ten w dużo większym stopniu dotyczy metod reguł prototypowych, gdyż transparentność modelu jest uzyskiwana właśnie poprzez reprezentację danych w postaci minimalnego zbioru prototypów gwarantujących dużą zdolność generalizacji.
Problem znajdowania optymalnej reprezentacji danych w postaci wektorów referencyjnych można podzielić na ich selekcję i optymalizację. Pierwsza grupa metod (selekcja) charakteryzuje się wyborem prototypów bezpośrednio jako wektorów zbioru treningowego, podczas gdy metody optymalizacji zajmują się optymalizacją położenia wektorów prototypowych poprzez minimalizację pewnego kryterium. W przypadku metod optymalizacji nie jest wymagane by położenie prototypu odpowiadało konkretnemu przypadkowi ze zbioru treningowego. W metodach optymalizacji dają się wyznaczyć dwie główne strategie - optymalizacja bezpośrednia, gdzie minimalizacji podlega bezpośrednio błąd klasyfikacji danych oraz optymalizacja pośrednia, gdzie wykorzystuje się algorytmy nienadzorowanej analizy danych, jak metody analizy skupień (klasteryzacji).
Inną możliwością podziału metod znajdywania prototypów jest podział na metody przyrostowe, w których rozpoczyna się poszukiwanie wektorów wzorcowych od pustego zbioru $\mathbf{P} = \emptyset$ dodając do niego nowe przypadki maksymalizujące pewne kryterium oraz metody redukcyjne. Rozpoczynają one działanie od pełnego zbioru prototypów $\mathbf{P} = \mathbf{T}$ usuwając z niego wektory zbędne, które nie wpływają na poprawę generalizacji algorytmu. Cechą charakterystyczną metod przyrostowych jest fakt, iż zawiera się w niej cała grupa metod optymalizacji.
W niniejszej rozprawie algorytmy doboru optymalnego zbioru wektorów referencyjnych podzielono według pierwszego kryterium na algorytmy selekcji prototypów, algorytmy klasteryzacji danych służące do wyboru przypadków referencyjnych oraz metody optymalizacji w znaczeniu optymalizacji bezpośredniej.

6.1 Metody selekcji prototypów

Rodzina algorytmów selekcji prototypów jest pierwszą grupą metod użytych w celu polepszenia jakości klasyfikacji algorytmu k-NN. W systemach reguł prototypowych

ma ona kilka istotnych właściwości bardzo przydatnych podczas interpretacji wyników. Jak już zostało wspomniane powyżej, metody selekcji wybierają wektory referencyjne spośród elementów zbioru treningowego, co oznacza że zbiór wybranych prototypów stanowi podzbiór realnych przypadków, które wystąpiły w przeszłości. Ma to szczególne znaczenie dla pewnych zastosowań w tym medycznych, gdzie prototyp stanowi konkretny przykład pacjenta z określonym schorzeniem lub prawnych w prawie precedensowym, gdzie wybrany prototyp jest konkretną sprawą, która miała miejsce w przeszłości.
W metodach selekcji prototypów można wyznaczyć dwa główne zadania, kondensacji danych - metody, w których usuwane są wektory „wewnętrzne", leżące daleko od granicy decyzji nie mające wpływu na jakość klasyfikacji oraz metody filtrów lub edycji usuwające wektory odstające (ang. outliers) [137]. Metody selekcji prototypów stosowane w praktyce zawierają często obydwa typy algorytmów, usuwając zarówno wektory odstające, jak i kondensując dane, co w znaczny sposób wpływa na uzyskaną jakość rezultatów.
Ze względu na nieciągłość funkcji decyzyjnej klasyfikatora k-NNnajczęściej stosowanym rozwiązaniem są metody przeszukiwania. W literaturze problem selekcji wektorów referencyjnych był studiowany przez wiele ośrodków naukowych, dlatego też poniżej zamieszczono listę najpopularniejszych metod pomijając wiele z nich jak Variable Similarity Metric zaproponowaną przez Lowe, Model Class Selection Brodleya, Typical Instance Based Learning stanowiącą pomysł Zanga, RISE Domingosa [39] i wiele innych jak [93]. Wiele metod selekcji prototypów, w szczególności bazujących na algorytmach genetycznych/ ewolucyjnych można znaleźć również na stronach projektu Keel [92]

6.1.1 Metoda ENN

Pierwszym bardzo skutecznym i prostym algorytmem selekcji prototypów jest zaproponowana przez Wilsona [183] reguła edycji ENN (ang. editing nearest neighbor rule). Metoda ta usuwa wszystkie wektory stanowiące szum w zbiorze danych treningowych. Dla każdego wektora wyznaczanych jest k najbliższych sąsiadów spośród zbioru $\mathbf{T}$, które użyte są do głosowania. Jeżeli wynikiem głosowania k sąsiadów jest błędna klasa, wówczas wektor taki zostaje oznaczony do usunięcia $\mathbf{P} = \mathbf{T} \backslash \bar{\mathbf{T}}$, gdzie $\bar{\mathbf{T}}$ jest zbiorem wektorów przeznaczonych do usunięcia. Rezultatem działania tego algorytmu jest więc usunięcie wektorów odstających oraz wektorów brzegowych, a jedyną wartością nastawną jest k (autor zaleca $k = 3$). Działanie algorytm przedstawia schemat (1).

6.1.2 Modyfikacje algorytmu ENN

Modyfikacją algorytmu ENN jest metoda RENN (ang. repeated ENN), gdzie algorytm ENN jest wielokrotnie powtarzany aż do momentu, w którym żaden z wektorów nie jest już usuwany w wyniku działania algorytmu ENN (schemat (2)). Kolejną modyfikacją jest algorytm *All* k-NN, gdzie porównywane są wyniki dla różnych wartości parametru k. Obydwie te metody zostały zaproponowane przez Tomeka i opisane w [166].

6.1.3 Metoda kondensacyjna CNN

Algorytm CNN (ang. condensed nearest neighbor rule) jest pierwszą metodą kondensacyjną zaproponowaną przez Harta [71] w 1967 roku. Metoda ta należy do grupy przyrostowych. Rozpoczyna ona od losowo wybranego wektora jako prototypu $\mathbf{P}$, następnie w pętli klasyfikuje pozostałe przypadki i jeżeli któryś zostaje błędnie

Schemat 1 Schemat algorytmu ENN

Require: $\mathbf{T}$
  $m \leftarrow sizeof(\mathbf{T})$;
  $rem_i \leftarrow 0$;
  for $i = 1 \ldots m$ **do**
    $\bar{C}(\mathbf{x}_i) = k\text{-NN}((\mathbf{T} \setminus \mathbf{x}_i), \mathbf{x}_i)$;
    if $C(\mathbf{x}_i) \neq \bar{C}(\mathbf{x}_i)$ **then**
      $rem_i = 1$;
    end if
  end for
  for $i = 1 \ldots m$ **do**
    if $rem_i == 1$ **then**
      $\mathbf{T} = \mathbf{T} \setminus \mathbf{x}_i$
    end if
  end for
  return $\mathbf{P}$

Schemat 2 Schemat algorytmu RENN

Require: $\mathbf{T}$ $flaga \leftarrow$ **true**
  while falga **do**
    $flaga \leftarrow$ **false**
    $\bar{\mathbf{P}} = \text{ENN}(\mathbf{P}, \mathbf{T})$
    if $\bar{\mathbf{P}} \neq \mathbf{P}$ **then**
      $flaga \leftarrow$ **true**
    end if
  end while
  return $\mathbf{P}$

sklasyfikowany przez aktualny zbiór prototypów jest on do niego dodawany $\mathbf{P} = \mathbf{P} \cup \mathbf{x}$. Procedura ta jest powtarzana aż wszystkie wektory zostają sklasyfikowane poprawnie. Wynikiem działania CNN jest mała liczba wektorów referencyjnych, mniejsza niż w algorytmach z serii ENN. Wadą tego rozwiązania jest jednak występowanie wszystkich wektorów odstających i brzegowych, co nie wpływa na poprawę generalizacji modelu wynikowego. Inną wadą rozwiązania CNN jest losowa inicjalizacja algorytmu. Powoduje to, że wielokrotny start może prowadzić do nieporównywalnych zbiorów wynikowych, na co również wpływa losowa kolejność prezentacji wektorów podczas procesu uczenia. Działanie algorytmu przedstawia schemat (3).

6.1.4 Metody redukcyjne RNN oraz DROP1-5

Gates zaproponował algorytm RNN (ang. reduced nearest neighbor rule), którego zasada działania jest również podobna do CNN, z tą jednak różnicą, że jest to algorytm należący do grupy redukcyjnych. Rozpoczyna on od pełnego zbioru wektorów referencyjnych $\mathbf{P} = \mathbf{T}$ i usuwa nieużyteczne wektory ze zbioru $\mathbf{P}$. Kryterium użytym przez Gatesa jest nie pogarszanie wyników klasyfikacji zbioru $\mathbf{T}$. Algorytm ten jest niestety obliczeniowo bardziej kosztowny niż CNN w procesie uczenia, jednakże jego wynikiem jest mniejszy rozmiar $\mathbf{P}$, co przynosi wymierne korzyści w procesie klasyfikacji przypadków testowych. Inną zaletą tego algorytmu jest usuwanie wektorów odstających,

Schemat 3 Schemat algorytmu CNN

Require: T
  $m \leftarrow sizeof(\mathbf{T})$
  $\mathbf{p}_1 \leftarrow \mathbf{x}_1$
  $flaga \leftarrow$ **true**
  while flaga **do**
    $flaga \leftarrow$ **false**
    for $i = 1 \ldots m$ **do**
      $\bar{C}(\mathbf{x}_i) = k\text{-NN}(\mathbf{P}, \mathbf{x}_i)$
      if $\bar{C}(\mathbf{x}_i) \neq C(\mathbf{x}_i)$ **then**
        $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{x}_i$;
        $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$
        $flaga \leftarrow$ **true**
      end if
    end for
  end while
  return P

a wynikowy zbiór prototypów stanowi zbiór wektorów brzegowych stanowiących, leżących bezpośrednio blisko granicy.
Algorytm DROP stanowi natomiast grupę metod, zaś ich nazwa pochodzi od rozwinięcia skrótu *decremental reduction optimization procedure* i jest rozwinięciem metod RT1-3 (ang. reduction technique) zaproponowanych przez Wilsona i Martineza [185, 184].
Pierwszy z algorytmów - Drop1 jest bardzo podobny do algorytmu RNN z tą różnicą, iż dokładność jest walidowana na zbiorze **P** w zamian za **T**, jak to miało miejsce w oryginale. Algorytm ten usuwa zarazem wektory odstające, jak i szum - jednak wadą tego rozwiązania była możliwość usuwania całych klastrów danych lub w skrajnych przypadkach całych klas. Dlatego też autorzy zaproponowali modyfikację nazwaną Drop2, która porządkowała kolejność prezentacji wektorów w zależności od odległości do najbliższego wroga (najbliższego wektora z przeciwną etykietą klasy) oraz wrócili do walidacji dokładności algorytmu realizowanej na pełnym zbiorze treningowym **T**.
W metodzie Drop3 Wilson i Martinez zastosowali dodatkowo filtr poprzedzający proces kondensacji danych w postaci algorytmu ENN. Dzięki takiemu rozwiązaniu zredukowana została liczba prototypów oraz zmniejszono tendencję do przeuczania sie systemu. Kolejny algorytm Drop4 dodatkowo poprawiał zdolności generalizacji modelu poprzez modyfikację algorytmu ENN tak, by sprawdzał czy filtracja danego wektora nie wpłynie negatywnie na klasyfikację pozostałych przypadków. Ostatni z algorytmów Drop5 jest bardzo podobny do algorytmu Drop2 z tą jednak różnicą, iż zamieniona została strategia porządkowania wektorów od najbliższego przeciwnika do najdalszego. Taka strategia spowodowała wygładzenie granicy decyzji, usuwając w pierwszej kolejności wektory leżące bezpośrednio na granicy pomiędzy klasami.

6.1.5 Metody GE oraz RNG

Obydwa algorytmy stanowią rozwinięcie metod kondensacji danych pozostawiając jedynie wektory aktywnie tworzące granicę decyzji pomiędzy klasami. Bazują one na diagramach Voronoi, które jednoznacznie definiują granicę pomiędzy poszczególnymi prototypami. Obydwa te algorytmy wybierają jedynie te wektory, które sąsiadują w

sensie Voronoi z wektorami z przeciwnych klas, dzięki czemu granica decyzji pozostaje niezmieniona w stosunku do oryginalnego algorytmu 1NN.
Głównym problemem występującym przy wykorzystaniu diagramów Voronoi jest złożoność obliczeniowa. Algorytmy GE (ang. gabriel editing) ma złożoność rzędu $O(n^3)$ i bazuje na sprawdzeniu dla każdej pary zbioru prototypów $\mathbf{p}_a$ i $\mathbf{p}_b$ zależności (6.1).

$$\bigvee_{a \neq b \neq c} D^2(\mathbf{p}_a, \mathbf{p}_b) > D^2(\mathbf{p}_a, \mathbf{p}_c) + D^2(\mathbf{p}_b, \mathbf{p}_c) \tag{6.1}$$

Podobnie algorytm RNG (ang. relative neighborhood graph) [10] dla każdej pary wektorów $\mathbf{p}_a$ oraz $\mathbf{p}_b$ posługuje się zależnością (6.2).

$$\bigvee_{a \neq b \neq c} D(\mathbf{p}_a, \mathbf{p}_b) \geq \max(D(\mathbf{p}_a, \mathbf{p}_c), D(\mathbf{p}_b, \mathbf{p}_c)) \tag{6.2}$$

6.1.6 Metoda IBL

IBL (ang. instance base learning) [2] stanowi ogólne podejście do problemu eksploracji danych składające się z pięciu różnych metod. IB1 jest metodą bazową będącą podstawą do dalszych rozwiązań. Jest to algorytm k-NN, w którym zaimplementowano narzędzia do analizy brakujących wartości oraz jako wstępne przetwarzanie danych dodano moduł normalizacji atrybutów ciągłych. W algorytmie IB2 (bazując na IB1) zastosowano podobne podejście do CNN z tą jednak różnicą, że realizowane jest pojedyncze przejście przez zbiór danych treningowych. Kolejną modyfikacją jest algorytm IB3, będący również algorytmem przyrostowym jak IB2 - jednakże zastosowano w nim zasadę „zaczekaj i sprawdź" (ang. wait and see), aby wyznaczyć zbiór wektorów, które będą gwarantowały poprawną klasyfikację i które należy zachować jako prototypy $\mathbf{P}$. W celu wyznaczenia akceptowalnych przypadków referencyjnych w algorytmie IB3 posłużono się przedziałem ufności zdefiniowanym jako:

$$AC_{up|low} = \frac{p + z^2 \Big/ 2n \pm z\sqrt{p(1-p)/n + z^2/4n^2}}{1 + z^2/n} \tag{6.3}$$

gdzie p oznacza prawdopodobieństwo sukcesu w n próbach, natomiast z jest poziomem ufności.
Wg. założeń algorytmu prototyp jest akceptowalny, jeżeli dolna granica jego dokładności (z poziomem ufności 0.9) jest większa niż górna granica częstości występowania jego klasy oraz prototyp jest usuwany, jeżeli górna granica jego dokładności jest niższa (ze współczynnikiem ufności 0.7) niż dolna granica częstości występowania jego klasy.
Dokładna analiza algorytmów IB1-3 dokonana przez autora D.Aha wskazała na małą dokładność uzyskiwaną przy analizie danych zawierających dużą liczbę cech, dlatego też autor zaproponował dodatkowe dwa algorytmy IB4 oraz IB5 mające rozwiązać ten problem.
Ogólne podejście IBL rozwijane przez Aha znalazło szerokie zainteresowanie również wśród innych autorów i doczekało się wielu modyfikacji, w tym zaproponowany przez Wilsonsa i Martineza algorytm Integrated Instance-Based Learning [182], gdzie autorzy dodali narzędzia automatycznego doboru parametrów i ważenia decyzji lub podejście Hullermeier, gdzie w oryginalnym algorytmie wykorzystano teorię posybilistyczną (ang. possibilistic theory) [75]

6.1.7 Metody selekcji losowej, genetycznej i ewolucyjnej

Duża grupa metod selekcji prototypów bazuje na metodach stochastycznego przeszukiwania, czego najprostszym przykładem są algorytmy selekcji losowej i Monte Carlo [107, 101, 160, 137] występujące w dwóch typowych formach. Obydwie formy bazują na skończonej liczbie iteracji z, w których niezależnie losowanych jest l prototypów, a jako funkcja celu używany jest błąd klasyfikacji klasyfikatora k-NN(zwykle $k = 1$).
Typowe podejścia do problemu losowania determinujące wspomniane dwie formy można opisać jako:

- prototypy losowane jednocześnie z całego zbioru treningowego (schemat (4))
- prototypy losowane niezależnie z każdej z klas z osobna, co determinuje konieczność określenia $l_{1..c}$ niezależnych wartości liczby prototypów osobno dla każdej klasy (schemat (5))

Schemat 4 Algorytm Monte Carlo w wersji typu 1

Require: $\mathbf{T}$, l, $maxit$
  for $i = 1 \ldots maxit$ **do**
    $\mathbf{tP} \leftarrow Rand(\mathbf{T}, l)$
    $tacc \leftarrow Acc(1NN(\mathbf{tP}, \mathbf{T}))$
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$
      $\mathbf{P} \leftarrow \mathbf{tP}$
    end if
  end for
  return $\mathbf{P}$

Schemat 5 Algorytm Monte Carlo w wersji typu 2

Require: $\mathbf{T}$, $\mathbf{l}$, $maxit$
  for $i = 1 \ldots maxit$ **do**
    $\mathbf{tP} = \emptyset$
    for $j = 1 \ldots c$ **do**
      $\mathbf{tP} \leftarrow \mathbf{tP} \cup Rand(\mathbf{T}_{C_j}, l_j)$
    end for
    $tacc \leftarrow Acc(1NN(\mathbf{tP}, \mathbf{T}))$
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$
      $\mathbf{P} \leftarrow \mathbf{tP}$
    end if
  end for
  return $\mathbf{P}$

Innym przykładem metod stochastycznych jest algorytm Random Mutation Hill Climbing zaproponowany przez Skala [160]. W podejściu tym każdy prototyp zakodowany jest w postaci podłańcuchów binarnych o długości $\log_2 m$, a całkowita długość łańcuch binarnego dla l prototypów wynosi więc $l \log_2 n$. Podczas każdej iteracji

algorytmu wspinaczki następuje permutacja jednego bitu. Wynikiem działania jest więc zbiór bitów reprezentujący l prototypów dający największe dopasowanie, definiowane jako dokładność klasyfikatora 1NN. Uproszczoną budowę algorytmu przedstawia schemat (6).

Schemat 6 Schemat algorytmu RMHC

Require: $\mathbf{T}$, l, $maxit$
  $m \leftarrow sizeof(\mathbf{T})$
  $st \leftarrow Bin(l \log_2 m)$
  $SetBit()$
  for $i = 1 \ldots maxit$ **do**
    $\mathbf{tBP} \leftarrow GetProto(\mathbf{T}, st)$
    $tacc \leftarrow Acc(1NN(\mathbf{tBP}, \mathbf{T}))$
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$
      $\mathbf{P} \leftarrow \mathbf{tBP}$
    end if
    $st \leftarrow PermuteBit(st)$
  end for
  return $\mathbf{P}$

Problemem selekcji prototypów metodami genetycznymi i ewolucyjnymi zajmowało się wielu autorów, gdzie na uwagę zasługuje Keel Project, na którego stronach internetowych [92] można znaleźć dużą rodzinę wspomnianych metod. Zagadnienie to było również poruszane w pracach Kunchevy m.in. [107], gdzie badania autorki wskazują, iż w problemach analizy małych zbiorów danych proste metody selekcji losowej przewyższają dokładnością bardziej zaawansowane metody genetyczne.

6.1.8 Encoding Length

Bardzo ciekawym rozwiązaniem selekcji wektorów referencyjnych jest algorytm zaproponowany przez Cameron Jonesa [23]. Prace tego autora skoncentrowane były na poprawieniu algorytmu Ahy IB3 oraz Skali Random Mutation Hill Climbing (RMHC) w skrócie opisane w rozdziale 6.1.7 przy wykorzystaniu heurystyki Encoding Length Heuristic (ELH) - szczegóły heurystyki zostały opisane w rozdziale 6.5. Heurystyka ELH została tutaj użyta jako funkcja dopasowania, która łączy w sobie dokładność klasyfikacji z liczbą prototypów. Główną bazą algorytmu była metoda IB3, gdzie po wstępnej selekcji zbiór $\mathbf{P}$ był oczyszczany metodami zachłannego przeszukiwania wykorzystując ELH do oceny prototypów, tak by zbiór wynikowy zawierał jedynie istotne wektory prototypowe (algorytm nazwany Pre/All, zwany również przez Wilsona i Martineza ELGrow). Inną modyfikacją był algorytm Explore rozszerzający Pre/All poprzez RMHC jako końcowy etap przetwarzania.
Na uwagę zasługuje fakt, iż metoda ta przez wielu autorów [184, 23, 82] była oceniana jako jedna z najlepszych, gwarantując bardzo mały zbiór prototypów z maksymalnie dużą dokładnością klasyfikacji.

6.2 Metody klasteryzacji w poszukiwaniu prototypów

Powszechnie stosowanymi metodami znajdywania wektorów referencyjnych opisywanymi między innymi [107, 25, 101, 17, 42, 137] są różne metody analizy skupień zwane również metodami klasteryzacji. W odróżnieniu od metod selekcji prototypów opisanych w poprzednim rozdziale 6.1, metody klasteryzacji znajdują prototypy nie będące przypadkami wektorów zbioru treningowego.
Grupowanie danych jest dużą niezależną dziedziną rodziny problemów inteligencji obliczeniowej, łączącą różne podejścia do problemu klasteryzacji. Ogólnie metody te można podzielić na kilka grup:

- Modele Gaussowskie
- Metody rozmytej i posibilstycznej klasteryzacji
- Sieci neuronowe (Samoorganizujące się mapy Kohonena)
- Metody hierarchiczne

Modele Gaussowskie wywodzą się z ogólnej probabilistycznej teorii Bayesa i należą do grupy metod parametrycznej estymacji rozkładów prawdopodobieństwa. Najczęściej stosowanym rozwiązaniem jest tutaj odwzorowywanie rozkładu danych (ich rozkładu prawdopodobieństwa $p(x)$) przy pomocy kombinacji rożnych rozkładów normalnych [61, 173].
Alternatywnym rozwiązaniem równie często spotykanym są metody bazujące na teorii zbiorów rozmytych oraz teorii posibilistycznej. Najlepszym przykładem tych metod jest rozmyty algorytm C-średnich (ang. fuzzy C-means - FCM) oraz posibilistyczny algorytm C-średnich (Possibilistic C-means - PCM) [147, 73]. Obydwa te algorytmy w wersjach podstawowych szukają zbiorowisk danych poprzez kombinacje Gaussowskich funkcji przynależności pokrywających każdy z atrybutów. Porównanie wyżej opisanych metod klasteryzacji zawarte w pracy Fenga [63] wskazuje, iż rezultaty uzyskane z obydwu rodzin metod (rozmytych i probabilistycznych) dają porównywalne rezultaty, zaś wadą obydwu metod jest problem inicjalizacji algorytmów.
Innym powszechnie stosowanym narzędziem pozwalającym na poszukiwanie zgrupowań danych są metody kwantyzacji wektorów (ang. vector quantization - VQ) lub samoorganizujące się sieci Kohonena (ang. self organizing maps - SOM) [98] należące do rodziny sieci neuronowych. Obydwie grupy metod bazują na zasadach „zwycięzca bierze wszystko” (ang. winner takes all - WTA) lub zwycięzca bierze większość (ang. winner takes most WTM) [133]. Przydatność obydwu opisanych metod do problemu uczenia klasyfikatora k-NNbyła testowana m.in. przez Kuncheve i Bezdeka [8] dając wyniki porównywalne z innymi metodami selekcji prototypów.
Ostatnią grupę metod klasteryzacji stanowią metody hierarchiczne, które posiadają bardzo istotną cechę, gwarantującą jednoznaczność rozwiązania. Metody hierarchiczne bazują na budowie drzewa - dendogramów, łącząc stopniowo leżące najbliżej siebie wektory w jeden klaster. Problem łączenia wektorów w klastry jest w opisywanej metodzie niezależną kwestią wpływającą na kształt uzyskanych zbiorowisk. Metody hierarchiczne były jednymi z pierwszych metod poszukiwania prototypów, czego przykładem może być algorytm Changa [25] z 1974 roku. Metoda ta rozpoczynała od pełnego zbioru prototypów $\mathbf{P} = \mathbf{T}$ i iteracyjnie łączyła pary najbliżej siebie leżących wektorów z $\mathbf{P}$ zamieniając je pojedynczym przypadkiem wyznaczanym jako ważona

średnia lub jak w zmodyfikowanym algorytmie Changa (ang. modified chang method - MCA) wyznaczając położenie zwykłą średnią. Algorytm Changa był rozwijany również przez innych, czego wynikiem był uogólniony algorytm Changa (ang. generalized MCA) wykorzystujący prawdziwy aglomeracyjny algorytm hierarchicznego grupowania [121]. Pękalska i inni w [137], jak również Kunchewa [101, 103] przedstawiają dwa różne podejścia do znajdowania prototypów przy wykorzystaniu ogólnych metod klasteryzacji - grupowanie danych całego zbioru treningowego z pominięciem informacji o klasach nazywane również klasteryzacją z powtórnym etykietowaniem (ang. clustering and relabeling) oraz klasteryzacją każdej z klas niezależnie. Kunchewa w [106] nazywa również wspomniane metody odpowiednio metodami uczenia nadzorowanego finalnego (ang. post-supervised) i uczenia nadzorowanego poprzedzającego (ang. pre-supervised).

Pierwsza z metod może zostać przedstawiona jak na schemacie (7). gdzie krok 4

Schemat 7 Metoda klasteryzacji z powtórnym etykietowaniem

Require: $\mathbf{T}$, l
Ensure: $l > c$
  $\mathbf{Cl} \leftarrow Cluster(\mathbf{T}, l)$
  $\mathbf{P} \leftarrow \emptyset$
  for i=1...l **do**
    $\mathbf{p}_i \leftarrow \text{mean}(\mathbf{cl}_i)$
    $C(\mathbf{p}_i) \leftarrow \arg\max(C(\mathbf{cl}_i))$
    $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{p}_i$
  end for
  return $\mathbf{P}$

może w zależności od wybranego rozwiązanie zostać zrealizowany jako etykietowanie ostre lub rozmyte. W etykietowaniu ostrym klasa, do której należy prototyp określana jest jako pojedyncza wartość uzyskana poprzez większościowe głosowanie wszystkich wektorów należących do danego klastra. W podejściu drugim - rozmytym etykietowaniu - każdy prototyp związany jest z każdą z klas c poprzez odpowiedni wektor wag $\mathbf{w} = [w_1, w_2, \ldots, w_c]$, w którym w_i jest wagą i-tej klasy. Wagi dla poszczególnych klas są z reguły definiowane jako $m_{c_i}^{l_j}/m^{l_j}$,gdzie $m_{c_i}^{l_j}$ oznacza liczbę wektorów znajdujących się w klastrze l_j przypisanych do c_i, a m^{l_j} jest całkowitą liczbą wektorów zawartych w klastrze l_j. Przedstawione *rozmyte* rozwiązanie ma pewną istotną wadę związaną ze złożonością procesu wnioskowania, co w znaczny sposób redukuje transparentność systemu ograniczając użyteczność tej metody w systemach reguł prototypowych.
Druga metoda niezależnej klasteryzacji jest dużo bardziej praktyczna, upraszczając proces wnioskowania, często również gwarantując większą dokładność systemu. Algorytm tej metody przedstawia schemat (8)

Wadą wykorzystania klasteryzacji do pozyskania prototypów jest, podobnie jak w przypadku części metod stochastycznych, konieczność określenia liczby prototypów. Problem ten jest niezmiernie istotny i w znaczący sposób wpływa na przejrzystość zbudowanego modelu. W szczególności dotyczy przypadku niezależnego grupowania, gdyż wymaga określenia c niezależnych parametrów osobno dla każdej z klas. Kwestia optymalizacji liczby prototypów została opisana w rozdziale 6.5. Ogólnie jednak należy zauważyć bardzo dobre wyniki uzyskiwane z opisanych metod, charakteryzujące się oczekiwanym balansem pomiędzy dokładnością a liczbą prototypów. Bardzo częstym rozwiązaniem jest również stosowanie metod klasteryzacji jako wstępnej inicjalizacji poprzedzającej nadzorowaną optymalizację położenia prototypów. Przykładowe wyniki

Schemat 8 Metoda niezależnego grupowania klas

Require: **T**, l
  $\mathbf{P} \leftarrow \emptyset$
  for i=1...c **do**
    $\mathbf{Cl} \leftarrow Cluster(\mathbf{T}_{C_i}, l_i)$
    $\mathbf{p} \leftarrow \text{mean}(\mathbf{Cl}_i)$
    $C(\mathbf{p}) \leftarrow i$
    $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{p}$
  end for
  return **P**

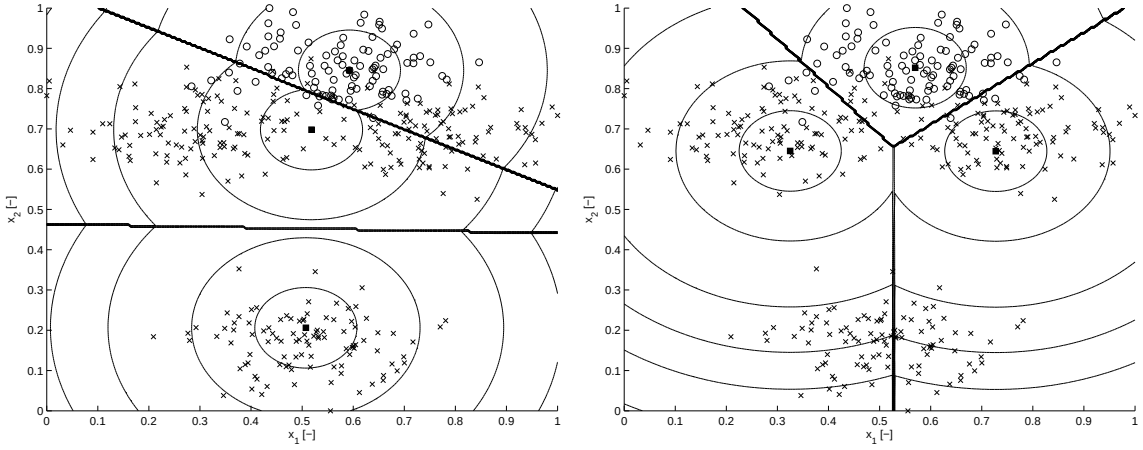
porównawcze można znaleźć w [17].

6.3 Nowa metoda poszukiwania prototypów bazująca na warunkowym grupowaniu danych

Wyznaczanie położenia prototypów poprzez klasyczne metody klasteryzacji pomija sporą część informacji o wzajemnym położeniu klas, co w znacznym stopniu ogranicza możliwości uzyskanego klasyfikatora, zmniejszając zarówno jego zdolność uogólniania, jak i transparentność. Analiza położenia prototypów wskazuje, iż częstym rezultatem jest nadmiarowa liczba wektorów, które nie biorą udziału w procesie klasyfikacji. Przykładem tego jest rys.(6.1a), gdzie jeden z prototypów jest bezużyteczny, nie wpływając na wynik klasyfikacji. Częściowym rozwiązaniem takiego problemu mogą być metody bazujące na diagramach Voronoi (jak GE lub RNG [82]) usuwające zbędne prototypy. Innym rozwiązaniem są tutaj metody semi-nadzorowanej klasteryzacji, charakteryzujące się możliwością wykorzystania informacji o względnym rozkładzie klas (rys.(6.1b)). Ogólna grupa metod klasteryzacji warunkowej wykorzystuje zewnętrzną zmienną zdefiniowaną dla każdego wektora wejściowego, tworząc tak zwany kontekst grupowania opisujący siłę oddziaływania pomiędzy wektorem wejściowym a pewnym zewnętrznym parametrem. Warunkowe (kontekstowe) grupowanie było rozwijane przez wielu autorów w tym m.in. przez W. Pedrycza [136, 135] i J. Łęskiego [114, 115], a znalazło zastosowanie również w problemach uczenia radialnych sieci neuronowych [134].
Rozwiązanie wykorzystujące warunkową klasteryzację do poszukiwania prototypów dla systemów reguł prototypowych zostało zaproponowane przez autora i innych w [17] i bazuje na algorytmie warunkowego rozmytego grupowania c-średnich (ang. Conditional Fuzzy C-means - CFCM). Jednak, co należy podkreślić może zostać również wykorzystane do dowolnej innej metody realizującej kontekstowe grupowanie danych.

6.3.1 Algorytm warunkowego rozmytego grupowania c-średnich

Jedną z podstawowych metod kontekstowego grupowania danych jest algorytm warunkowego rozmytego grupowania c-średnich (ang. conditional fuzzy C-means



(a) Selekcja prototypów algorytmem FCM (b) Selekcja prototypów algorytmem CFCM

Rysunek 6.1: Porównanie wyników selekcji prototypów metodami grupowania typu FCM oraz CFCM

CFCM), który bazuje na minimalizacji funkcji kosztu zdefiniowanej jako (6.4).

$$J_j(\mathbf{U},\mathbf{P}) = \sum_{i=1}^{l_i}\sum_{k=1}^{m_j}(u_{ik})^p \left\|\mathbf{x}_k - \mathbf{p}_i\right\|_A^2 \tag{6.4}$$

gdzie l_j oznacza liczbę klastrów (prototypów) wyznaczonych dla danej klasy j, m_j to liczba wektorów treningowych z tej klasy ($m = \sum_j m_j$), c to całkowita liczba klas, $p > 1$ to parametr określający stopień rozmycia, zaś $\mathbf{U} = (\mathbf{u_{ik}})$ jest macierzą o rozmiarach $c \times m_j$, której elementy $u_{ik} \in [0,1]$ określają stopień przynależności k-tego wektora do i-tego klastra. Aby rozwiązać tak zdefiniowaną funkcję kosztu konieczne jest by macierz $\mathbf{U}$ spełniała trzy kryteria:
1^o każdy wektor x_k przynależy do i-tego klastra w stopniu z przedziału [0,1]:

$$\bigvee_{1\le i\le c}\bigvee_{1\le k\le m} u_{ik} \in [0,1] \tag{6.5}$$

2^o suma stopni przynależności k-tego wektora x_k do wszystkich klastrów równa jest wartości kontekstu f_k

$$\bigvee_{1\le k\le m}\sum_{i=1}^{c} u_{ik} = f_k \tag{6.6}$$

3^o żaden klaster nie jest pusty

$$\bigvee_{1\le i\le c} 0 < \sum_{k=1}^{m} u_{ik} < m \tag{6.7}$$

Przedstawiona funkcja celu (6.4) z uwzględnieniem opisanych warunków jest iteracyjnie minimalizowana zgodnie z formułami (6.8) (6.9):

$$\bigvee_{1\le i\le c} \mathbf{p}_i = \sum_{k=1}^{m}(u_{ik})^p \mathbf{x}_k \left[\sum_{k=1}^{m}(u_{ik})^p\right]^{-1} \tag{6.8}$$

$$\underset{\substack{1\le i\le c\\ 1\le k\le m}}{\forall} u_{ik} = f_k \left[\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{p}_i\|}{\|\mathbf{x}_k - \mathbf{p}_j\|} \right)^{2/(p-1)} \right]^{-1} \tag{6.9}$$

6.3.2 Kontekst grupowania

W metodach prototypowych bazujących na algorytmie k-NNnajlepszymi wektorami referencyjnymi $\mathbf{P}$ są wektory leżące w pobliżu granicy między klasami, dzięki czemu możliwe jest zbudowanie optymalnej granicy decyzji z minimalną liczbą prototypów. Założenie to prowadzi do wniosku, iż algorytm grupowania powinien być skoncentrowany na klasteryzacji podprzestrzeni leżącej w pobliżu granicy międzyklasowej. Wymagane jest więc stworzenie takiej wartości f_k, która spełni powyższe wymagania.
Możliwym rozwiązaniem, opisanym przez autora w [17] jest zależność (6.10)

$$w_k = \sum_{j, C(\mathbf{x}_j)=C(\mathbf{x}_k)} \|\mathbf{x}_k - \mathbf{x}_j\|^2 \left[\sum_{l, C(\mathbf{x}_l)\neq C(\mathbf{x}_k)} \|\mathbf{x}_k - \mathbf{x}_l\|^2 \right]^{-1} \tag{6.10}$$

gdzie uzyskana wartość parametru w_k wymaga normalizacji, aby rezultat był określony w przedziale [0,1] (6.11)

$$w_k = \left(w_k^- \min_k (w_k) \right) \left(\max_k (w_k) - \min_k (w_k) \right)^{-1} \tag{6.11}$$

Parametr w_k określa zatem położenie wektora względem jego sąsiadów i po znormalizowaniu przyjmuje wartości bliskie 0 dla wektorów treningowych w pobliżu dużych jednorodnych środków klas, zaś bliskie 1 dla wektorów x_k leżących daleko od przedstawicieli tej samej klasy, lecz blisko wektorów klasy przeciwnej. Innymi słowy duże wartości w_k odpowiadają położeniu w pobliżu granicy, a maksymalne występują dla wektorów odstających.
Tak zdefiniowany znormalizowany parametr x_k wymaga filtracji określającej interesujący nas obszar klasteryzacji, stanowiący kontekst grupowania dla algorytmu CFCM. Filtracja jest często interpretowana również jako przypisywanie rozmytej wartości lingwistycznej do zmiennej w_k w znaczeniu zbiorów rozmytych. W metodzie opisanej przez autora użyto filtra Gaussowskiego (6.12)

$$f_k = \exp\left(\frac{-(w_k - \mu)^2}{\sigma^2} \right) \tag{6.12}$$

gdzie parametr μ określa położenie przestrzeni klasteryzacji, zaś σ to szerokość tego obszaru.
Przeprowadzone badania wskazały, iż najlepszymi wartościami filtra są $\mu = 0.6 - 0.8$ i $\sigma = 0.6 - 0.9$, determinując poprawne działanie algorytmu dla szerokiej skali problemów.

6.4 Optymalizacja położenia prototypów

Nadzorowana optymalizacja położenia prototypów dla reguł prototypowych klasyfikatora k-NNjest istotnym elementem procesu uczenia. Standardowe metody

selekcji, jak i optymalizacji z wykorzystaniem algorytmów grupowania nie są w stanie uzyskać tak optymalnego położenia prototypów, jakie możliwe jest do uzyskania z wykorzystaniem w pełni nadzorowanej optymalizacji gradientowej dobierającej położenie prototypów tak by zminimalizować pewną funkcję kosztu, najczęściej przedstawianą jako błąd klasyfikacji. Najpowszechniej stosowanym rozwiązaniem jest tu grupa algorytmów Learning Vector Quantization (LVQ) zaproponowanych przez Kohonena [98].
Algorytm LVQ należy do grupy metod sieci neuronowych i ma istotną wadę (podobnie jak większość metod sieci neuronowych) - jest bardzo czuły na inicjalizację wag, w tym przypadku rozumianą jako inicjalizację położenia prototypów. Wada ta istotnie wpływa na powtarzalność uzyskiwanych rezultatów, dlatego też bardzo dobrą praktyką metod LVQ jest inicjalizacja położenia prototypów poprzez metody selekcji lub klasteryzacji danych. Takie rozwiązanie było analizowane m.in. przez Jankowskiego i Grochowskiego w [83], dając bardzo dobre rezultaty. Inną istotną wadą oryginalnego algorytmu LVQ jest brak zdolności określania optymalnej liczby prototypów, co pociąga za sobą konsekwencje manualnego określania liczby neuronów (prototypów).
Klasyczny algorytm LVQ bazuje na zasadzie „zwycięzca bierze wszystko" i wykorzystuje metodę optymalizacji zbliżoną do gradientowej, gdyż optymalizowana funkcja celu jest nieciągła. Podstawowy algorytm LVQ doczekał się wielu modyfikacji - sam Kohonen zaproponował cztery wersje algorytmu oznaczone LVQ1, LVQ2 (LVQ2.1), LVQ3 oraz OLVQ, a w literaturze spotykać można szereg innych wersji działających w trybie wsadowym (ang. batch) lub ciągłym (ang. online), uwzględniających ważenie cech oraz szereg innych właściwości [85, 86, 87, 88, 24].

6.4.1 Algorytmy rodziny LVQ

LVQ1 jest algorytmem bazowym dla całej grupy, który iteracyjnie optymalizuje położenie wektorów referencyjnych, zwanych również wektorami kodującymi (ang. codebook vectors).
Proces uczenia może przebiegać w dwóch trybach: wsadowym, gdy aktualizacja wektorów kodujących odbywa się po prezentacji całego zbioru uczącego oraz w trybie ciągłym, gdy po prezentacji pojedynczego wektora treningowego następuje modyfikacja położenia najbliższego prototypu zgodnie z formułą (6.13).

$$\begin{aligned} \mathbf{p}_i(z+1) &= \mathbf{p}_i(z) - \alpha(z)(\mathbf{x}_j - \mathbf{p}_i(z)) \\ \mathbf{p}_i(z+1) &= \mathbf{p}_i(z) + \alpha(z)(\mathbf{x}_j - \mathbf{p}_i(z)) \end{aligned} \tag{6.13}$$

$$i = \arg\min_{a=1\ldots l}(\|\mathbf{x}_j - \mathbf{p}_a\|) \tag{6.14}$$

gdzie i oznacza najbliższy prototyp do j-tego wektora trenującego $\mathbf{x}$ według zależności (6.14), $\alpha(i)$ jest współczynnikiem uczenia - monotonicznie malejącą funkcją przyjmującą wartości pomiędzy $[0, 1]$, a z oznacza aktualną iterację.
Przeprowadzona w ramach prac innych badaczy analiza porównawcza oraz wyniki eksperymentów wskazują, iż uczenie w trybie ciągłym jest bardziej stabilne i mniej odporne na wektory odstające [133].

LVQ 2 (2.1) jest bardziej zaawansowanym algorytmem, który optymalizuje jednocześnie dwa wektory kodujące z przeciwnych klas, leżące najbliżej danego wektora

treningowego. Jeżeli w bezpośrednim sąsiedztwie danego wektora treningowego obydwa najbliższe prototypy są z tej samej klasy, algorytm pozostawia je bez zmian. Modyfikacja położenia prototypów odbywa się zgodnie z procedurą LVQ1, jednakże modyfikacja obydwu prototypów naraz prowadzi do uzyskania lokalnej optymalności w sensie Bayesowskim.
Algorytm LVQ2.1 posiada drobną modyfikację w stosunku do swego poprzednika, polegającą na uwzględnieniu tylko takich wektorów $\mathbf{x}$, które trafiają do okna o względnej szerokości w zdefiniowanego jako (6.15).

$$\min\left(\frac{d_1}{d_2}, \frac{d_2}{d_1}\right) > s \; ; \; s = \frac{1-w}{1+w} \tag{6.15}$$

gdzie d_1 i d_2 oznaczają odległości pomiędzy wektorem $\mathbf{x}$ a dwoma najbliższymi prototypami $\mathbf{p}$. W praktycznych zastosowaniach algorytmu LVQ2.x stosuje się przetwarzanie kaskadowe, gdzie działanie tej metody poprzedzone jest optymalizacją zrealizowaną w oparciu o LVQ1.

LVQ3 jest kolejnym rozszerzeniem metod rodziny LVQ, w którym analizowana jest sytuacja wystąpienia dwóch najbliższych prototypów do danego wektora treningowego, wszystkich o jednakowych etykietach. Proces adaptacji położenia prototypów odbywa się wówczas według zależności (6.16).

$$\mathbf{p}_i(z+1) = \mathbf{p}_i(z) + \varepsilon\alpha(z)(\mathbf{x}_j - \mathbf{p}_i(z)) \tag{6.16}$$

gdzie ε jest stałą zależną od szerokości okna w, która przyjmuje wartości pomiędzy 0.1 a 0.5. Rezultatem, a zarazem celem algorytmu LVQ3 jest zapewnienie nie tylko poprawnej klasyfikacji danych, ale również odtworzenie rozkładu punktów w klasach.

OLVQ1 jest ostatnim algorytmem z serii podstawowych metod zaproponowanych przez Kohonena. Modyfikacja zaproponowana w tej metodzie polega na uniezależnieniu współczynnika uczenia osobno dla każdego z prototypów - zależność (6.17). Według założeń, taka modyfikacja powinna prowadzić do przyspieszenia zbieżności algorytmu.

$$\begin{aligned}\mathbf{p}_i(z+1) &= \mathbf{p}_i(z) - \alpha_i(z)(\mathbf{x}_j - \mathbf{p}_i(z)) \\ \mathbf{p}_i(z+1) &= \mathbf{p}_i(z) + \alpha_i(z)(\mathbf{x}_j - \mathbf{p}_i(z))\end{aligned} \tag{6.17}$$

Odpowiednio $\alpha_i(z)$ oznacza współczynnik uczenia przypisany do i-tego wektora kodującego w iteracji z.

6.4.2 Modyfikacje algorytmów LVQ

LVQ-TC jest metodą zaproponowaną przez Ordorico w [131], w której modyfikacja położenia prototypu zależy nie tylko od wzajemnej odległości pomiędzy prototypem a wektorem treningowym, ale również od historii prototypu. Według autora, historia prototypu definiowana jest jako niezależna dla każdego z wektorów referencyjnych liczba modyfikacji jego położenia, określająca osobno ilość nagród i kar. Ostateczna postać modyfikacji prototypu przybiera wówczas formę (6.18).

$$\mathbf{p}_i(z+1) = \mathbf{p}_i(z) + \frac{\alpha(z)}{\sum \mathbf{q}_i}(\mathbf{x}_j - \mathbf{p}_i(z)) \tag{6.18}$$

gdzie $\sum \mathbf{q}_i$ oznacza ilość modyfikacji i-tego prototypu (suma wszystkich modyfikacji i-tego wektora ; $\mathbf{q}_i = [q_1, q_2, \ldots, q_c]^T$ jest liczba modyfikacji zrealizowanych poprzez wektory z k-tej klasy). Jednocześnie autor zaproponował kryterium oczyszczające - usuwające zbędne prototypy, porównując liczbę modyfikacji q_i ze stałą Q, określającą minimalną liczbę modyfikacji. Jeżeli prototyp nie był dostatecznie często modyfikowany jest on usuwany. Kolejnym rozszerzeniem algorytmu jest powtórne etykietowanie występujące w sytuacji, gdy prototyp i był w większości modyfikowany przez wektory z klasy k i jeżeli $k > Q$ - wówczas prototypowi takiemu przypisywana jest etykieta z klasy k, a on sam umiejscawiany jest w środku wektorów z klasy k, które wpływały na jego przesunięcie.

Algorytm DSM (ang. decision surface mapping) [8] jest prostą modyfikacją algorytmu LVQ, w której położenie prototypu jest modyfikowane jedynie poprzez nałożenie kary. Innymi słowy autorzy Geva and Sitte sugerują, by na zmianę położenia prototypu wpływały jedynie wektory błędnie klasyfikowane. DSM często prowadzi do lepszych rezultatów klasyfikacji niż tradycyjny algorytm LVQ, jednakże metoda ta wydaje się być mniej stabilna.
W przedstawionych wynikach rozprawy użyto kolejnej modyfikacji, w której zostały określone dwa niezależne współczynniki uczenia osobno dla nagradzania i karania, dzięki czemu możliwe stało się uzyskanie kompromisu pomiędzy stabilnością a dokładnością klasyfikacji. Najczęściej przyjętą zależnością pomiędzy współczynnikiem kary a nagrody jest $\alpha_{kary} = 2\alpha_{nagrody}$

Dynamic LVQ [7] charakteryzuje się możliwością automatycznego doboru liczby prototypów. Algorytm realizowany jest w dwóch krokach, w pierwszym uruchamiany jest klasyczny algorytm LVQ, zaś w drugim dodawany jest nowy prototyp. Obydwa te kroki wykonywane są do momentu zatrzymania opadania funkcji kosztu (definiowanej jako liczba błędów). Każdy nowy prototyp umiejscawiany jest w obszarze Voronoi, wynikającym z obecnego rozkładu prototypów, w którym rejestrowana jest największa liczba błędów. Położenie nowego prototypu wyznaczane jest wówczas jako średnia błędnie klasyfikowanych wektorów należących do tego obszaru Voronoi pochodzących z klasy C_i (6.19).

$$\mathbf{p} = \frac{1}{h_{C_i}} \sum_{k=1}^{h_{C_i}} \mathbf{x}_k \tag{6.19}$$

gdzie h_{C_i} to liczba błędnie klasyfikowanych wektorów z klasy C_i.
Jako ostatni krok uruchamiany jest algorytm oczyszczania usuwający te prototypy, które nie biorą udziału w klasyfikacji.

6.5 Wyznaczanie optymalnej liczby prototypów

W systemach reguł prototypowych przejrzystość modelu jest uzyskiwana m.in. poprzez możliwie jak najmniejszą liczbę prototypów. Jednakże proces selekcji lub ekstrakcji prototypów wymaga optymalizacji wielokryterialnej, gdzie niezbędny jest kompromis pomiędzy dokładnością modelu a liczbą prototypów oraz liczbą cech. Typowy proces wyznaczania położenia prototypów dla reguł prototypowych typu k-NNjest szeregowym

procesem przetwarzania o dwóch lub trzech krokach. W pierwszym etapie wyznaczane jest wstępne położenie prototypów, jako rezultat systemów selekcji lub klasteryzacji. Następnie, w kroku drugim położenie prototypów podlega optymalizacji by zwiększyć zdolność dyskryminacji klas. Krok trzeci, nie zawsze stosowany, służy oczyszczeniu zbioru prototypów poprzez usunięcie zbędnych wektorów referencyjnych nie biorących udziału w klasyfikacji. Ostatni etap realizowany jest zwykle metodami kondensacji danych (CNN, RNN) lub metodami bazującymi na diagramach Voronoi (GE, RNG)[82], w taki sposób by usunąć wektory wewnętrzne-redundantne.
Część z metod redukcji zbioru wektorów referencyjnych pozwala na automatyczny dobór ich liczby. Dla szeregu jednak metod bazujących na grupowaniu, algorytmach genetycznych czy też metodach optymalizacji niezbędne jest manualne określenie liczby wektorów prototypowych.
Niektóre metody selekcji prototypów bazujące na algorytmach genetycznych posiadają tak skonstruowaną funkcję celu, która zawiera informację o jakości klasyfikacji w stosunku do liczby wektorów referencyjnych, pozwalając jednocześnie na ich automatyczny dobór. Podobnie często stosowaną praktyką w przypadku niezależnej klasteryzacji klas jest wykorzystanie metody optymalizacji doboru liczby klastrów, opisywane między innymi w pracy Stąpor [164] oraz Bischopa [14]. Jednak rezultaty uzyskiwane tą drogą są dalekie od optymalnych. Powodem tego jest brak bezpośredniej korelacji jakości grupowania oraz dokładności klasyfikacji, czego przykładem może być rys.(6.1). Dlatego też w zagadnieniu tym stosowane są algorytmy przeszukiwania. Najczęściej problem ten dotyczy metod, w których konieczna jest równoległa optymalizacja liczby prototypów dla każdej z klas z osobna.
Pewną alternatywą optymalizacji liczby prototypów jest zaproponowany przez autora *algorytm wyścigu* (ang. racing algorithm) [17] pozwalający na wielokrotne przyspieszenie optymalizacji liczby prototypów, przy możliwym minimalnym spadku jakości klasyfikacji.
Osobnym, acz bardzo istotnym problemem jest walidacja modelu zabezpieczająca przed przeuczeniem. Częściowo realizuje się to poprzez metody statystycznej kontroli złożoności lub weryfikację w oparciu o walidację krzyżową - dokładniej temat ten będzie analizowany w rozdziale 8.

6.5.1 Algorytmy przeszukiwania

Typowe algorytmy przeszukiwania można podzielić na tzw. nienadzorowane, ślepe metody przeszukiwania (jak metody szukania wszerz lub wgłąb) oraz metody nadzorowane, heurystyczne (jak przeszukiwanie metodą wspinaczki, wiązką lub inne). Zagadnienie optymalizacji liczby prototypów jest typowym problemem optymalizacji heurystycznej, czego przykładem może być praca [8].
Najczęściej stosowanym rozwiązaniem jest algorytm wspinaczki (ang. hill-climbing), którego zastosowanie do optymalizacji liczby prototypów można przedstawić w postaci schematu (9), gdzie PNN($\mathbf{l}$) to prototypowy algorytm k-NNo $\mathbf{l}$ prototypach, a $Validate(M, \mathbf{T})$ to funkcja walidująca model M w oparciu o zbiór $\mathbf{T}$.

Metoda ta niestety nie gwarantuje znalezienia globalnego optymalnego rozwiązania, jednakże ze względu na dużą prostotę i szybkość jest powszechnie stosowana. Metoda przeszukiwania wiązką jest również często stosowana, pozwalając na pełniejsze przeszukanie, co pozwala często na uzyskiwanie lepszych rezultatów oraz uniknięcie ekstremów lokalnych. Jej wadą jest jednak dużo większy nakład obliczeniowy oraz

Schemat 9 Optymalizacja liczby prototypów metodą wspinaczki

Require: **T**, **l**
Ensure: **l** $\leftarrow [1, 1, \ldots, 1]$
  $acc \leftarrow Validate(\mathrm{P}k\text{-NN}(\mathbf{l}), \mathbf{T})$
  $flaga \leftarrow$ **true**
  tlb $\leftarrow$ **l**
  while $flaga$ **do**
    $flaga \leftarrow$ **false**
    for i=1...c **do**
      $tla_i \leftarrow l_i + 1$
      $tacc \leftarrow Validate(\mathrm{PNN}(\mathbf{tla}), \mathbf{T})$
      if tacc>acc **then**
        $flaga \leftarrow$ **true**
        $acc \leftarrow tacc$
        tlb $\leftarrow$ **tla**
      end if
    end for
    l $\leftarrow$ **tlb**
  end while
  return **l**

większe zapotrzebowanie na zasoby systemowe, co dla dużych problemów badawczych stanowi poważne ograniczenie.

6.5.2 Nowy *algorytm wyścigu* optymalizacji liczby prototypów

Wadą algorytmów szukania jest konieczność przechodzenia poszczególnych stanów w grafie, co znacząco wpływa i determinuje koszt nakładu obliczeniowego. Najprostszą więc metodą redukcji złożoności obliczeniowej jest takie sformułowanie problemu, które redukuje liczbę węzłów lub pozwala na przeszukanie drzewa pomijając pewną liczbę węzłów i przeszukując tylko te, które rokują najlepszą poprawę funkcji celu. *Algorytmu wspinaczki* w procesie optymalizacji liczby wektorów referencyjnych wymaga dla każdego ze stanów wykonania c (gdzie c oznacza liczbę klas) procesów uczenia algorytmu bazowego (np. CFCM+LVQ) podczas, gdy dla *algorytm wyścigu* wymagane jest pojedyncze wywołanie uczenia.
Własność tę osiągnięto bazując na obserwacji, iż w algorytmach selekcji prototypów widoczna jest tendencja, w której dokładność klasyfikacji pojedynczej klasy wpływa na całkowitą dokładność klasyfikacji. Na tej podstawie można więc wysnuć wniosek, że maksymalizacja poprawy całkowitej dokładności klasyfikacji możliwa jest do osiągnięcia poprzez poprawę dokładności najgorzej klasyfikowanej klasy. Założenia takie realizowane są przez *algorytm wyścigu* (ang. racing algorithm) - schemat (10), gdzie nowy prototyp dodawany jest do klasy o najgorszej dokładności klasyfikacji. Iteracyjne dodawanie nowego prototypu do klasy o największym błędzie (C_i^{err}) powoduje więc wzrost jej dokładności klasyfikacji, co z kolei podnosi całkowitą zdolność dyskryminacji systemu. Przeprowadzone badania wskazują jednak, że w pewnych wyjątkowych sytuacjach uzyskane rozwiązania mogą być nieoptymalne i dodanie nowego prototypu do danej klasy może nawet obniżyć całkowitą dokładność klasyfikacji. Zjawisko takie występuje, gdy dodawanie nowego prototypu nie powoduje wzrostu dokładności

klasyfikacji danej klasy, co może występować np. w przypadku klasycznych metod grupowania danych. Jednak zastosowanie *algorytmu wyścigu* wraz z grupowaniem warunkowym (CFCM) prowadzi do bardzo dobrych rezultatów, co dowodzą wyniki przedstawione w rozprawie. Powodem tego jest fakt, iż każdy nowo dodany wektor referencyjny bierze istotny udział w procesie klasyfikacji, poprawiając dokładność klasyfikacji klasy, do której prototyp został dodany.
W algorytmie tym błąd klasyfikacji klasy C_i^{err} jest niezależnie od funkcji optymalizowanej. Zarówno dla dokładności opisanej wyrażeniami (2.1), jak i (2.2) wybór klasy do której prototyp należy dodać określany jest jako maksymalny błąd wyrażony $C_i^{err} = \frac{m_i^{err}}{m_i}$, gdzie m_i jest liczbą wektorów w klasie i, a m_i^{err} to liczba błędów w tej klasie. Ocena błędu klasyfikacji najczęściej dokonywana jest w procesie walidacji krzyżowej. Kolejną korzyścią *algorytmu wyścigu* jest omijanie ekstremów lokalnych. Typowe,

Schemat 10 Optymalizacja liczby prototypów algorytmem wyścigu

Require: **T**, **l**, *maxit*
Ensure: $\mathbf{l} \leftarrow [1, 1, \ldots, 1]$
  $acc \leftarrow 0$
  tl $\leftarrow$ **l**
  for $i = 1 \ldots maxit$ **do**
    $tacc, C^{err} \leftarrow Validate(\mathrm{PNN}(\mathbf{tl}), \mathbf{T})$
    $id \leftarrow \arg\max(C^{err})$
    $tl_{id} \leftarrow tl_{id} + 1$
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$
      l $\leftarrow$ **tl**
    end if
  end for
  l $\leftarrow$ **tlb**
  return **l**

proste metody przeszukiwania jak *algorytm wspinaczki* kończą proces optymalizacji po osiągnięciu pierwszego ekstremum, czyli w sytuacji gdy zwiększenie liczby prototypów w dowolnej z klas nie powoduje zwiększenia dokładności klasyfikacji. *Algorytm wyścigu* omija to ograniczenie, gdyż każdorazowo dodaje nowy prototyp do gorzej klasyfikowanej klasy.

6.5.3 Funkcje kryterialne

Klasyczna funkcja kryterialna

Najprostszą i najczęściej stosowaną funkcją kryterialną w metodach przeszukiwania oraz wyścigu jest dokładność (2.1) oraz dokładność zbalansowana (2.2). Niestety poważną wadą tak zdefiniowanej funkcji celu jest brak informacji o stopniu skomplikowania oraz złożoności modelu. Może to prowadzić do niepotrzebnego wzrostu liczby prototypów, co negatywnie wpływa na zdolności interpretacji uzyskanych reguł.

Funkcje kryterialne uwzględniające liczbę prototypów

Alternatywnym i bardzo prostym rozwiązaniem pozbawionym powyższej słabości, a uwzględniającym liczbę prototypów jest propozycja Kunchevy [107]. Metoda ta polega

na liniowej kombinacji błędu klasyfikacji oraz względnej liczby prototypów (6.20).

$$J(\mathbf{P}) = (1 - Acc) + \alpha \frac{l}{m} \tag{6.20}$$

gdzie α jest współczynnikiem balansu pomiędzy liczbą prototypów a dokładnością klasyfikacji. Większa wartość α zwiększa koszt związany z liczbą prototypów, ograniczając tym samym wpływ dokładności klasyfikacji, zaś dla $\alpha = 0$ uzyskiwana jest klasyczna postać funkcji kosztu jako jedynie dokładność klasyfikacji.
Inną możliwością definicji funkcji kosztu, która w większym stopniu dąży do kompromisu pomiędzy liczbą prototypów a dokładnością klasyfikacji jest (6.21)

$$J(\mathbf{P}) = (1 - Acc)^2 + \left(\alpha \frac{l}{m}\right)^2 \tag{6.21}$$

Dzięki takiemu zapisowi powierzchnia rozwiązań jest wypukła, co powoduje, że skrajne przypadki w których bardzo duża liczba prototypów daje niewielki wzrost dokładności lub bardzo mała liczba prototypów dająca duży błąd są karane, a znajdywane rozwiązanie to średnia liczba prototypów przy średnio małym błędzie klasyfikacji.

Statystyczne metody oceny złożoności

Interesującą funkcję kosztu do optymalizacji dokładności i liczby prototypów zaproponował w swojej pracy Cameron-Jones w [23]. Funkcja ta bazuje na wskaźniku teorioinformacyjnym - na zasadzie minimalnej długości kodu lub bardziej precyzyjnie na heurystyce *encoding length*. Zaproponowana funkcja przyjmuje postać (6.22)

$$J(m, l, m^{err}) = F(l, m) + l \log_2(c) + F(m^{err}, m - l) + m^{err} \log_2(c - 1) \tag{6.22}$$

gdzie $F(l, m)$ jest kosztem kodowania l wektorów przez m instancji.

$$F(l, m) = \log^* \left(\sum_{j=0}^{l} \frac{m!}{j!(m-j)!} \right) \tag{6.23}$$

gdzie $\log^*$ jest sumą dodatnich czynników $\log_2$. Przedstawiona funkcja celu została użyta m.in. w algorytmie RMHC (ang. random mutation hill climbing) z bardzo dobrymi rezultatami, uzyskując bardzo dobrą dokładność przy jednoczesnej małej liczbie prototypów.
Innym spotykanym rozwiązaniem jest minimalizacja ryzyka strukturalnego (ang. structural risk minimization, SRM) [170] bazująca na wymiarze VC. Przykładowe rozwiązanie uwzględniające SRM do optymalizacji zostało opisane w [33].

6.6 Porównanie metod selekcji prototypów

Niezmiernie istotną kwestią jest wybór spośród dużej grupy przedstawionych powyżej metod, tych które w największym stopniu spełniają kryteria stawiane systemom reguł prototypowych. Od algorytmów wymagane jest więc uzyskanie maksymalnej zdolność generalizacji przy minimalnej liczbie prototypów. W tym celu dokonano porównania metod selekcji i optymalizacji prototypów na realnych zbiorach danych,

których opis można znaleźć w rozdziale 12. W przeprowadzonych testach porównano powszechnie znane metody opisane w literaturze z zaproponowanym przez autora algorytmem wykorzystującym parę - warunkowe rozmyte grupowanie danych wraz z algorytmem LVQ o niesymetrycznych współczynnikach nagrody i kary (CFCM+LVQ). Dodatkowo dla algorytmu CFCM+LVQ zamieszczono wyniki dla dwóch metody doboru liczby prototypów bazujących na *algorytmie przeszukiwania* (s-CFCM+LVQ) oraz opracowanym przez autora *algorytmem wyścigu* (rCFCM+LVQ). Wyniki porównania przedstawiono w tabeli (6.1). Do realizacji porównania wykorzystano wyniki prac Jankowskiego oraz Grochowskiego [82, 67] jak również obliczenia własne z wykorzystaniem pakietów GhostMiner [84] oraz stworzonego przez autora systemu SBPS. Zamieszczone w tabeli wyniki są średnią dokładnością oraz odchyleniem standartowym jak również średnią względną liczbą prototypów uzyskanymi z dziesięciokrotnego testu krzyżowego. W tabeli

6.7 Podsumowanie wyników

Uzyskane wyniki porównawcze wskazują na dużą skuteczność grupy algorytmów bazujących na metodzie długości kodowania (ang. encoding length), do której należą ELGrow oraz Explore - tym samym uwydatnia się ich przydatność do realizacji systemów reguł prototypowych typu k-NN. Szczególnie dobre rezultaty uzyskano algorytmem Explore, który obok małej liczby prototypów charakteryzował się dużą dokładnością. Natomiast metody ENN, RENN, AllKNN oraz CNN cechowały się bardzo dużą liczbą wyznaczonych wektorów referencyjnych, co negatywnie wpływa na zdolność interpretacji uzyskanych rezultatów. Podobne wnioski dotyczą metod GE oraz RNGE, które w porównaniu z metodami ELGrow i Explore nadal osiągają niezadawalającą liczbę wyznaczonych prototypów. W porównaniu do innych metod stosunkowo dobre rezultaty uzyskano dla metod z rodziny DROP1-5 oraz algorytmu IB3. Dodatkowym atutem tych metod jak i wspomnianych wyżej jest wybór wektorów będących realnymi przypadkami zbioru treningowego.
Część algorytmów z uwagi na brak wbudowanych metod optymalizacji liczby prototypów (LVQ, RMHC, MC1) posiadała na stałe zdefiniowaną liczbę prototypów równą 1 na klasę. W przypadku tych metod najlepsze rezultaty osiągnął algorytm LVQ, do wad którego należy fakt, iż znalezione prototypy nie są wektorami zbioru treningowego. Podobny problemy dotyczą również zaproponowanych w pracy algorytmów CFCM oraz CFCM+LVQ. Należy jednak podkreślić ich dużą skuteczność. Zarówno sam algorytm wykorzystujący grupowanie danych CFCM, jak i jego modyfikacja łącząca warunkowe grupowanie danych z algorytmem LVQ (CFCM+LVQ) wykazały bardzo dużą skuteczność, co więcej dla większości zbiorów uzyskały najlepsze rezultaty. W przypadku obydwu algorytmów liczba prototypów wyznaczana była dwoma różnymi metodami (wspinaczki i wyścigu) celem ich wzajemnego porównania. Uzyskane rezultaty są podobne, jednak widoczna jest niewielka przewagą średniej dokładności klasyfikacji algorytmu wyścigu (r_CFCM+LVQ). Zostało to jednak okupione średnio większą liczbą prototypów oraz większym rozrzutem ich liczby w poszczególnych walidacjach.

Dane:	Wyrostek robaczkowy		Rak piersi		Choroby wątroby	
Metoda	l/m	1-NN	l/m	1-NN	l/m	1-NN
$l = m$	100.0 ± 0.0	83.00 ± 84.41	100.0 ± 0.0	94.86 ± 4.16	100.0 ± 0.0	62.29 ± 6.37
ENN	84.65 ± 1.70	89.03 ± 7.84	96.26 ± 0.35	96.65 ± 2.28	64.80 ± 2.08	58.91 ± 9.06
RENN	84.62 ± 1.71	88.35 ± 8.74	95.85 ± 0.39	96.57 ± 2.06	56.59 ± 2.54	58.90 ± 10.28
AllKNN	75.38 ± 2.22	87.65 ± 8.54	92.71 ± 0.63	96.80 ± 2.21	35.05 ± 11.22	60.63 ± 12.42
CNN	39.42 ± 3.09	71.67 ± 13.52	10.92 ± 0.81	91.25 ± 3.25	63.80 ± 1.13	61.73 ± 8.72
DROP1	5.88 ± 1.83	78.02 ± 22.16	1.47 ± 0.57	87.56 ± 15.27	9.79 ± 4.05	60.36 ± 11.78
DROP2	7.24 ± 1.63	83.70 ± 12.02	3.34 ± 0.79	94.61 ± 2.87	25.60 ± 1.54	60.31 ± 5.04
DROP3	7.09 ± 1.24	85.95 ± 9.39	3.00 ± 0.64	96.08 ± 2.35	18.78 ± 2.64	63.77 ± 10.57
DROP4	7.28 ± 1.37	86.02 ± 10.26	3.32 ± 0.70	95.75 ± 2.69	25.35 ± 1.82	60.58 ± 5.56
DROP5	7.59 ± 1.05	87.25 ± 9.06	4.5 ± 0.90	95.14 ± 2.43	25.31 ± 3.35	64.34 ± 4.28
GE	43.56 ± 2.97	76.29 ± 12.70	46.79 ± 3.71	94.93 ± 2.50	83.38 ± 1.39	60.82 ± 6.42
RNGE	36.95 ± 3.10	78.12 ± 11.78	10.79 ± 0.71	95.42 ± 2.04	75.17 ± 1.04	61.45 ± 4.02
IB3	4.539 ± 2.46	79.30 ± 14.11	4.27 ± 0.67	96.14 ± 2.52	11.11 ± 1.66	58.51 ± 5.79
ELH	15.33 ± 4.55	87.58 ± 11.85	3.19 ± 0.66	91.66 ± 4.53	47.34 ± 4.95	62.65 ± 8.97
ELGrow	1.42 ± 0.62	80.67 ± 8.82	0.32 ± 0.0	92.16 ± 4.11	0.32 ± 0.00	57.98 ± 0.89
Explore	1.63 ± 0.58	81.89 ± 8.35	0.32 ± 0.0	95.13 ± 2.56	0.45 ± 0.17	57.38 ± 2.27
LVQ	2.1 ± 0.01	85.28 ± 10.40	0.32 ± 0.0	92.86 ± 2.69	0.64 ± 0.0	64.34 ± 6.97
RMHC	2.1 ± 0.01	85.12 ± 9.01	0.32 ± 0.0	96.39 ± 2.05	0.64 ± 0.0	55.37 ± 6.94
CFCM	2.1 ± 0.00	81.06 ± 17.54	0.37 ± 0.08	97.37 ± 1.78	0.81 ± 0.35	57.08 ± 5.74
r-CFCM+LVQ	2.1 ± 0.00	86.61 ± 13.32	0.5 ± 0.05	97.66 ± 1.83	0.74 ± 0.16	65.55 ± 6.48
s-CFCM+LVQ	1.88 ± 0.00	84.70 ± 16.20	0.366 ± 0.16	96.36 ± 2.74	0.86 ± 0.29	65.82 ± 2.74
Dane:	Choroby serca		Cukrzyca		Irysy	
Metoda	l/m	1-NN	l/m	1-NN	l/m	1-NN
$l = m$	100.0 ± 0.0	74.39 ± 7.55	100.00 ± 0.0	69.95 ± 4.54	100.00 ± 0.00	96.28 ± 4.62
ENN	80.31 ± 1.24	79.61 ± 7.59	73.15 ± 0.99	74.88 ± 4.24	95.51 ± 0.87	94.88 ± 4.80
RENN	76.94 ± 1.42	79.27 ± 6.85	69.18 ± 1.14	74.50 ± 3.94	95.10 ± 0.81	95.16 ± 4.59
AllKNN	62.72 ± 1.86	78.48 ± 7.25	54.28 ± 1.16	74.41 ± 4.60	92.31 ± 1.11	94.94 ± 5.12
CNN	42.73 ± 2.07	70.9 ± 7.11	50.53 ± 1.32	64.24 ± 5.75	16.50 ± 2.28	90.93 ± 6.78
DROP1	4.95 ± 2.21	67.45 ± 10.93	6.76 ± 1.53	64.27 ± 6.62	5.49 ± 0.83	90.29 ± 7.26
DROP2	15.91 ± 2.76	76.95 ± 7.93	17.84 ± 1.69	69.75 ± 5.18	11.94 ± 1.62	93.53 ± 5.41
DROP3	12.46 ± 2.31	77.76 ± 7.51	12.99 ± 1.38	72.58 ± 4.68	11.96 ± 1.53	93.79 ± 5.76
DROP4	15.43 ± 2.47	77.87 ± 7.00	17.77 ± 1.92	72.30 ± 4.94	12.19 ± 1.47	93.32 ± 5.45
DROP5	16.52 ± 2.49	77.31 ± 6.97	18.99 ± 1.80	70.52 ± 4.61	9.99 ± 2.12	93.96 ± 5.56
GE	99.07 ± 0.58	76.33 ± 7.45	67.85 ± 1.27	64.17 ± 5.03	66.60 ± 2.00	94.69 ± 5.25
RNGE	55.17 ± 1.73	74.08 ± 8.6	59.50 ± 1.05	67.23 ± 4.99	23.29 ± 1.39	92.90 ± 5.59
IB3	11.16 ± 1.90	76.86 ± 7.15	7.82 ± 0.97	69.12 ± 4.99	18.07 ± 5.45	94.04 ± 6.35
ELH	17.02 ± 2.72	72.31 ± 8.08	22.21 ± 2.05	67.33 ± 4.75	9.64 ± 1.95	92.24 ± 5.68
ELGrow	0.73 ± 0.09	73.1 ± 8.31	0.32 ± 0.15	69.26 ± 6.22	2.05 ± 0.41	88.92 ± 7.31
Explore	0.73 ± 0.00	80.08 ± 7.2	0.32 ± 0.10	72.45 ± 5.62	1.95 ± 0.39	95.38 ± 4.41
LVQ	0.73 ± 0.0	80.86 ± 7.02	0.29 ± 0.0	72.52 ± 4.40	1.87 ± 0.0	90.56 ± 7.11
MC1	0.73 ± 0.0	79.0 ± 8.03	0.29 ± 0.0	69.78 ± 4.98	1.87 ± 0.0	89.63 ± 7.95
RMHC	0.73 ± 0.0	79.03 ± 6.29	0.29 ± 0.0	70.09 ± 5.70	1.87 ± 0.0	89.53 ± 7.70
CFCM	0.9 ± 0.19	83.47 ± 6.74	0.29 ± 0.0	73.32 ± 6.14	4.37 ± 1.02	95.33 ± 6.36
r-CFCM+LVQ	0.97 ± 0.4	83.12 ± 5.85	0.51 ± 0.23	75.52 ± 3.53	4.13 ± 0.69	96.00 ± 5.62
s-CFCM+LVQ	0.84 ± 0.29	73.05 ± 8.67	0.32 ± 0.09	7683 ± 4.90	2.60 ± 0.37	92.67 ± 5.84

Tablica 6.1: Wyniki porównawcze różnych metod selekcji prototypów

Rozdział 7

Selekcja prototypów dla reguł prototypowych progowych

7.1 Wstęp

Reguły prototypowe progowe (PTR), w odróżnieniu od reguł najbliższego sąsiada k-NNpokrywają przestrzeń danych wejściowych zbiorem wypukłych ograniczonych funkcji, bazujących na podobieństwie w oparciu o zależność $D(\mathbf{x}, \mathbf{p}) < \Theta$, gdzie Θ to wartość progu, a $\mathbf{p}$ to położenie prototypu. Taki zapis funkcji decyzyjnej pozwala na dużą łatwość interpretacji rezultatów klasyfikacji, definiując w przestrzeni wejściowej podprzestrzenie o określonej etykiecie klasy wyjściowej, podobnie do systemów klasycznych reguł ostrych. PTR, w zależności od doboru funkcji odległości, pozwalają na większą elastyczność w dopasowywaniu reguł do danych - możliwe nawet jest uzyskanie granicy decyzyjnej podobnej do wyników reguł k-NNczy też granicy liniowej, jeżeli wektor prototypowy $\mathbf{p}$ odsunięty jest daleko w przestrzeni wejściowej - można to zapisać jako $p_i \rightarrow \inf$ - wówczas granicę taką można traktować jako odcinkowo liniową lub jeżeli macierz wag $\mathbf{W}$ ważonej funkcji odległości $D(\mathbf{p}, \mathbf{x}; \mathbf{W})$ przyjmują proporcjonalnie duże wartości dla odpowiednich składowych.
Z uwagi na łatwość interpretacji pożądane jest, aby wartość wyjściowa pojedynczej reguły decydowała o wyjściu całości systemu regułowego. Dlatego też do realizacji PTR stosowane są trzy różne algorytmy:

- Ograniczonej energii Kulomba (ang. restricted coulomb energy, RCE)
- Heterogeniczne drzewa decyzji (ang. heterogeneous decision trees, HDT)
- Uporządkowana lista reguł prototypowych progowych (ang. ordered prototype threshold decision list, OPTDL)

7.2 Algorytm RCE

Najprostszym algorytmem, którego można użyć jako narzędzia do uzyskiwania reguł prototypowych progowych, jest algorytm RCE [61] zaproponowany przez Reilly'ea, Coopera oraz Erlbauma. RCE w odróżnieniu od k-NNzamiast doboru wartości k określa promień lub próg r używany do predykcji. Sam algorytm jest bardzo prosty i można go przedstawić w postaci schematów (11) oraz (12) .

Schemat 11 Algorytm RCE, funkcja ucząca

Require: $\mathbf{T}$, λ, ϵ
  $m \leftarrow \text{sizeof}(\mathbf{T})$
  for $i = 1 \ldots m$ **do**
    $th_i \leftarrow \min_{C(\mathbf{x}') \neq C(\mathbf{p}_i)} (D(\mathbf{p}_i, \mathbf{x}'))$ {znajdź najbliższego wroga do $\mathbf{p}_i$}
    $th_i \leftarrow \min(th_i - \epsilon, \lambda)$
  end for
  $\mathbf{P} \leftarrow [\mathbf{T}, th]$
  return $\mathbf{P}$

Schemat 12 Algorytm RCE, funkcja testująca

Require: $\mathbf{x}$, $\mathbf{P}$
  $m \leftarrow \text{sizeof}(\mathbf{P})$
  for $i = 1 \ldots m$ **do**
    $\mathbf{P}' = \text{find}\,(D(\mathbf{x}, \mathbf{p}_i) < th_i)$ {Znajdź wszystkie prototypy zawierające $\mathbf{x}$}
    $C(\mathbf{x}) = \max \text{hist}(\mathbf{P}')$ {Głosuj pośród $\mathbf{P}'$ na etykietę klasy dla wektora $\mathbf{x}$}
  end for

  return $C(\mathbf{x})$

W przedstawionym algorytmie każdy prototyp ma wyznaczoną wartość progu według zależności: *Jeżeli* $r' < D(\mathbf{p}, \mathbf{x}_e)$ *to* $r = r'$ *w przeciwnym razie* $r = D(\mathbf{p}, \mathbf{x}_e)$ gdzie r' jest stałą określającą maksymalną wartość progu określaną przez użytkownika, a $D(\mathbf{p}, \mathbf{x}_e)$ jest odległością pomiędzy wektorem prototypowym $\mathbf{p}$, a jego najbliższym wrogiem $\mathbf{x}_e$ (wektorem z przeciwnej klasy). Proces klasyfikacji polega zaś na głosowaniu wszystkich prototypów, które aktywowane są przez wektor testowy $C(\mathbf{x}) = \max_i hist(\sum(D(\mathbf{p}, \mathbf{x}) < r \& C(\mathbf{p}) = C_i))$. W praktyce jednak powyższy algorytm jest rzadko stosowany do ekstrakcji reguł ze względu na dużą liczbę prototypów $\mathbf{P} = \mathbf{T}$ co zmniejsza przejrzystość systemu. Jednak algorytm ten podobnie jak k-NNstanowi punkt wyjścia dla dużej grupy innych metod.

7.3 Heterogeniczne drzewa decyzji

Standardowe drzewa decyzji, jak C4.5 [143], CART [22] lub drzewo SSV [65] używają jedynie pojedynczego typu testu w celu podziału danych o postaci: $T(x_i < \Theta)$, który dzieli przestrzeń wejściową w zależności od wartości cechy x_i na dwie lub więcej gałęzi, pozwalając na uzyskanie zestawu reguł ostrych. Heterogeniczne drzewa decyzji przedstawione i opisane w [66], używają co najmniej dwóch jakościowo różnych typów testów. Dodanie nowego typu testu $T(D(\mathbf{x}, \mathbf{p}_k) < \Theta_k)$ bazującego na podobieństwie do prototypów powoduje utworzenie zestawu reguł prototypowych progowych, czyli lokalnych podziałów - podprzestrzeni (dla odległości Euklidesa przyjmujących kształt hipersfer). Proces tworzenia takiego drzewa w najprostszym przypadku zakłada, że wszystkie wektory treningowe są traktowane jako prototypy, czego wynikiem jest kwadratowa macierz odległości $D(\mathbf{p}_i, \mathbf{p}_j)$ o wymiarze $m \times m$, gdzie m jest liczbą wektorów zbioru treningowego. W kolejnych etapach tworzenia drzewa następuje wybieranie tylko tych prototypów jako węzłów, które maksymalizują daną funkcję kryterialną, aż do uzyskania maksymalnej dokładności klasyfikacji. Miarą odległości mogą być

tutaj dowolne funkcje mierzące odległość lub podobieństwo jak np. Gaussowskie funkcje jądrowe. Istotną cechą drzew heterogenicznych jest kombinacja tradycyjnych podziałów hiperpłaszczyznami prostopadłymi do wybranych atrybutów przestrzeni wejściowej (reguły ostre) oraz sferycznych, lokalnych podziałów wynikających z kształtu funkcji odległości (reguły prototypowe). Takie rozwiązania będące kombinacją reguł klasycznych - ostrych oraz prototypowych prowadzą do bardzo interesujących rezultatów. Choć proces poszukiwania odpowiednich prototypów staje się tutaj bardzo kosztowny, o złożoności $O(m^2)$ to korzyści są imponujące - pozwalając przykładowo na uzyskanie na zbiorze Wisconsin Breast Cancer dokładności rzędu 97.3% przy najprostszym możliwym opisie danych w postaci pojedynczej regule progowej [66].

7.4 Uporządkowana lista reguł prototypowych progowych

Rezultatem działania heterogenicznych drzew decyzji jest hierarchiczna struktura drzewiasta uzyskanych reguł. Alternatywą pozwalającą na łatwiejszą interpretację wyników jest płaska lista reguł pokrywających dane. Taka lista uzyskiwana jest poprzez zaproponowany przez autora w pracach [19, 15] algorytm OPTDL. W odróżnieniu od algorytmu HDT, OPTDL tworzy jedynie reguły prototypowe progowe. Dlatego też rezultatem działania tego algorytmu jest zbiór reguł prototypowych progowych w postaci zbioru prototypów, odpowiedniej metryki odległości oraz przypisanej każdemu prototypowi wartości progu $\mathbf{\Theta}$. Ze względu na maksymalizację zdolności generalizacji, algorytm dopuszcza możliwość wzajemnego nakrywania się reguł, dlatego też tworzona lista reguł jest uporządkowana. Do tworzenia reguł w algorytmie zastosowano kryteria stosowane w drzewach decyzji typu indeks Gini, zysk informacyjny (ang. information gain), względny zysk informacyjny (ang. information gain ratio) czy też kryterium SSV. Metoda OPTDL jest zbliżona do metod sekwencyjnego pokrywania, stopniowo pokrywając przestrzeń danych regułami prototyp-próg, zaczynając od najbardziej ogólnej reguły pokrywającej największą ilość przypadków, a kończąc na regułach szczegółowych - specjalizowanych dla małego zbioru wektorów. Ze względu na łatwość interpretacji, decyzję o etykiecie klasy wyjściowej podejmuje pojedyncza reguła. Dzięki takiemu procesowi ekstrakcji wiedzy możliwe jest pokrycie przestrzeni danych minimalną liczbą reguł oraz ułatwiony jest proces klasyfikacji poprzez testowanie reguł w odwróconej kolejności od najbardziej szczegółowej do najbardziej ogólnej (testowanie reguły następuje tylko wówczas, gdy reguła ją poprzedzająca nie jest spełniona). W algorytmie za najbardziej ogólną regułę klasyfikującą, równoważną nie spełnieniu żadnej z reguł (*else*) uznaje się większościowe głosowanie etykiety wektorów nie pokrytych żadną z listy reguł.

Algorytm OPTDL działa na zasadzie przeszukiwania metodą „pierwszy najlepszy" (ang. best first search), gdzie przestrzeń przeszukiwań stanowi zbiór potencjalnych prototypów, czyli pełny zbiór wektorów treningowych oraz odpowiednich wartości odległości do pozostałych wektorów w zbiorze $\mathbf{T}$. Dodatkowo dla ułatwienia procesu interpretacji w metodzie OPTDL dla problemów wieloklasowych stosuje się określenie kierunku reguły odpowiednio $D(\mathbf{x}, \mathbf{p}_k) < \Theta_k$ oraz $D(\mathbf{x}, \mathbf{p}_k) > \Theta_k$ określające czy reguła stanowi obszar wewnętrzny podprzestrzeni, czy też zewnętrzny (dla problemów dwuklasowych obydwa typy reguł są równoważne).

Schemat (13) przedstawia przykładowy mikrokod funkcji realizującej proces uczenia

listy reguł, gdzie dla uproszczenia założono problem dwuklasowy. Lista reguł tworzona jest dopóki nie zostanie osiągnięta maksymalna liczba reguł (*maxrul*) lub do momentu, gdy wszystkie wektory zostaną poprawnie sklasyfikowane. Następnie każdy z wektorów treningowych rozważany jest jako prototyp tworzący regułę. W tym celu wyznaczany jest wektor odległości d pomiędzy rozważanym wektorem x_i i pozostałymi wektorami zbioru treningowego $\mathbf{T}$, a uzyskane wartości odległości są sortowane narastająco. W celu minimalizacji złożoności obliczeniowej kryterium utworzenia progu (funkcja CalcCrit$(\cdot)$) obliczane jest jedynie dla przypadków, gdy sąsiednie w znaczeniu odległości wektory mają różne etykiety klas. Do oceny progu reguły wykorzystywane są również informacje o dotychczasowej jakości klasyfikacji wektorów $\mathbf{T}$, informacje te zapisywane są w binarnym wektorze *icor_lab* o rozmiarze m, który przyjmuje wartości 1 dla wektorów błędnie sklasyfikowanych. Spośród wszystkich analizowanych wektorów x_i zapamiętywany wraz z odpowiednim progiem jest ten przypadek, który maksymalizuje wartość funkcji kryterium CalcCrit$(\cdot)$. Założenie określające maksymalną liczbę reguł,

Schemat 13 Mikrokod algorytmu OPTDL

Require: $\mathbf{T}$, *maxrul*
  while $rul\_n < maxrul$ or $sum(icor\_lab) == 0$ **do**
    for $i = 1 \ldots m$ **do**
      $d \leftarrow Dist(\mathbf{x}_i, \mathbf{T})$
      $d \leftarrow Sort(d)$
      for $j = 1 \ldots m - 1$ **do**
        if $C(d_j) \neq C(d_{j+1})$ **then**
          $t\_cri \leftarrow$ CalcCrit$(d_{\leq j}, d_{>j}, icor_lab)$
          if $t\_cri > cri$ **then**
            $th \leftarrow (d_j + d_{j+1})/2$
            $proto \leftarrow i$
            $cri \leftarrow t\_cri$
          end if
        end if
      end for
    end for
    $\mathbf{P} \leftarrow \mathbf{P} \cup [proto, th]$
    $icor\_lab \leftarrow$ ApplyRules$(\mathbf{P}, \mathbf{T})$
    $rul\_n \leftarrow rul\_n + 1$
  end while
  return $\mathbf{P}$

niestety pozbawione jest możliwości kontroli zdolności generalizacji algorytmu, dlatego też w algorytmie OPTDL zastosowano wewnętrzną k-krotną walidację krzyżową, służącą ocenie zdolności uogólniania (podobnie jak w drzewie decyzji SSV [65, 66]). Podczas realizacji walidacji, dla każdego jej kroku, tworzona jest nowa lista reguł, przy czym pożądane optymalizowane kryterium jest sprawdzane dla każdej nowo dodanej reguły. W ostatecznym kroku tworzenia listy, jej rozmiar jest dobierany jako średnia dokładność wyników uzyskanych kolejno dla każdego poziomu listy, pomniejszona o jej odchylenie standardowe (rozdział 8.2). Ostatecznie więc na całym zbiorze treningowym tworzona jest lista Z reguł, (gdzie Z to liczba reguł wynikająca z wewnętrznej walidacji), które zapewniają maksymalną dokładność przy minimalnej wariancji wyników.

7.5 Poprawa kryterium podziału dla algorytmów drzew decyzji oraz OPTDL

Zarówno systemy drzew decyzji, jak również drzew heterogenicznych czy też algorytm OPTDL używają kryteriów statystycznych (jak np. indeksy Gini, SSV). Wszystkie one bazują na analizie liczebności przypadków, oceniając w ten sposób jakość podziałów dla tworzonej struktury wiedzy. Z uwagi na dyskretny charakter wartości liczebności wektorów dla poszczególnych podziałów możliwe jest występowanie impasów, objawiających się równością wartości funkcji kryterium statystycznego dla kilku podziałów. Typowe algorytmy drzew decyzji podejmują wówczas decyzję w sposób losowy, jednak przeprowadzona przez autora rozprawy analiza wskazuje, iż dla atrybutów ciągłych możliwe jest ominięcie problemu powstawania impasów. Jest to realizowane dla dowolnego indeksu statystycznego ($\mathrm{In}(f, x_{if}, x_{jf})$) poprzez uwzględnienie marginesu separowalności μ co można wyrazić jako (7.1)

$$\begin{gathered} \mu(x_{if}, x_{jf}) = \frac{\left|x_{if} - x_{jf}\right|}{\max(f) - \min(f)} \\ \mathrm{In}'(f, x_{if}, x_{jf}) = 2 \cdot \mathrm{In}(f, x_{if}, x_{jf}) + \mu(x_{if}, x_{jf}) \end{gathered} \tag{7.1}$$

Wówczas w kryterium $\mathrm{In}(\cdot)$ uwzględnia się znormalizowaną odległość (do przedziału [0,1]) pomiędzy parą wektorów x_{if},x_{jf}, która wyznacza miejsce podział danego atrybutu lub reguły f.

7.6 Porównanie metod reguł prototypowych progowych

Weryfikacji dokładności zaproponowanego w pracy algorytmu OPDL dokonano porównując wyniki klasyfikacji z wynikami uzyskanymi dla metod drzew heterogenicznych. Podczas realizacji testów porównawczych algorytm HDT był testowany w dwóch trybach - z pojedynczym kryterium (oznaczenie HDT), gdzie do budowy drzewa wykorzystano jedynie atrybuty wynikające z odległości, tym samym uzyskanym wynikiem była drzewiasta struktura reguł prototypowych progowych oraz w trybie drugim (HDT^{mix}), trybie heterogenicznym, gdzie zbiór atrybutów stanowił połączenie normalnego zbioru treningowego z atrybutami wynikającymi z wzajemnych odległości wektorów. Do realizacji algorytmu drzew heterogenicznych wykorzystano algorytm drzew decyzji bazujący na indeksie Gini. Procedura testowa obejmowała test dziesięciokrotnej walidacji krzyżowej algorytmów realizowany na zbiorach danych opisanych w rozdziale (12). Uzyskane wyniki średniej dokładności oraz odchylenie standardowe wraz z liczbą wyznaczonych reguł przedstawia tabela (7.6).

7.7 Podsumowanie wyników

Analiza uzyskanych rezultatów wskazuje, iż nie można jednoznacznie wskazać algorytmu dominującego. Każda z metod dla różnych zbiorów danych uzyskiwała najlepsze rezultaty. Nawet w przypadku dwóch wersji algorytmu HDT istniały czasami istotne różnice w dokładności. Należy tutaj zwrócić uwagę, iż dla zbiorów *jonosfera*, *wyrostek robaczkowy* oraz *choroby wątroby* lepsze rezultaty uzyskano stosując jedynie reguły

	HDT		HDT^{mix}		OPDL	
Zbiór danych	Dokładność	l	Dokładność	l	Dokładność	l
Cukrzyca	71.76 ± 5.10	2	74.87 ± 5.52	2	72.14 ± 5.36	1
Choroby serca	81.75 ± 6.83	1	81.75 ± 6.83	1	81.45 ± 6.66	1
Jonosfera	91.60 ± 8.76	3	90.09 ± 6.41	3	91.79 ± 6.23	7
Rak piersi	97.23 ± 2.10	1	97.23 ± 2.10	1	97.38 ± 2.14	1
Wyrostek robaczkowy	85.62 ± 11.60	1	83.80 ± 10.47	1	86.70 ± 14.92	4
Sonar	59.17 ± 14.53	4	61.12 ± 10.16	5	64.60 ± 11.71	5
Choroby wątroby	58.55 ± 7.53	2	57.68 ± 5.17	2	59.36 ± 9.40	5
Lancet	93.91 ± 2.56	6	93.48 ± 2.21	1	93.20 ± 3.28	5

Tablica 7.1: Porównanie wyników dla systemów reguł prototypowych progowych (HDT,OPTDL) oraz systemu heterogenicznego HDT^{mix}.

prototypowe progowe, nie zaś ich kombinację z regułami klasycznymi (HDT^{mix}). Chociaż często wzrost wartości średniej dokładności powodował też zwiększenie wartości odchylenia standardowego wyników. Różnice w przypadku algorytmów HDT mogą też być spowodowane problemem realizacji przeszukiwania, gdyż algorytm drzewa decyzji stosował metodę przeszukiwania typu *najpierw najlepszy*. Jej zastąpienie metodą *przeszukiwania wiązką* powinno ustabilizować wyniki, i podnieść średnią dokładność algorytmu HDT^{mix}, gdyż ze względu na dużą liczbę atrybutów rośnie tutaj znacznie problem przeszukiwania.
Wyniki uzyskane algorytmem OPDL są nieznacznie gorsze w stosunku do wyników algorytmów HDT, jednak zaletą algorytmu OPDL jest płaska struktura uzyskanych reguł, co w niektórych przypadkach może znacznie ułatwić proces interpretacji uzyskanych reguł. Uzyskane wyniki wskazują również, na nieznaczny wzrost liczby uzyskanych reguł, jednak dla niektórych zbiorów, jak *jonosfera* czy też *cukrzyca* algorytm OPDL pozwolił na uzyskanie najlepszych wyników dokładności w porównaniu z algorytmem HDT tworzącym jedynie reguły prototypowe progowe.

Rozdział 8

Selekcja cech i modeli

Głównym celem ekstrakcji wiedzy z danych jest taka jej reprezentacja, by była łatwa i zrozumiała dla człowieka. Dlatego też najczęściej wiedzę przedstawia się w postaci zbioru reguł, gdyż taki zapis wydaje się być najbardziej intuicyjny i powinien w spełnić własność łatwości interpretacji (prostota systemu). Można więc ponownie przytoczyć tutaj kryteria, które powinien spełniać dobrze zdefiniowany system regułowy - z jednej strony powinien przedstawiać informacje w postaci jak najmniejszego zbioru reguł, przy czym liczba przesłanek w poszczególnych regułach powinna być również minimalna zwiększając prostotę systemu, z drugiej zaś strony powinien charakteryzować się jak największą generalizacją i dokładnością. W przypadku reguł prototypowych (reguł-P) problem selekcji prototypów, a tym samym problem redukcji liczby reguł, został opisany w rozdziale 6. Jednakże spełnienie opisanego kryterium prostoty wymaga również rozważenia problemu selekcji cech, redukując tym samym liczbę przesłanek.
Niezależnym i bardzo istotnym problemem systemów inteligencji obliczeniowej jest poszukiwanie optymalnego modelu dla danego zagadnienia. Powszechnie znane twierdzenie *nie ma obiadu za darmo* (ang. No free lunch theorem) wskazuje na to, iż nie istnieje jeden najlepszy algorytm lub ich kombinacja, pozwalająca na znalezienie zawsze optymalnego rozwiązania. Dlatego też konieczne staje się znalezienie odpowiednich metod oceny oraz przeszukiwania - zwanych meta uczeniem (ang. meta learning), pozwalających na rozwiązanie takiego problemu. Innym istotnym zagadnieniem, również związanym z problemem meta uczenia, jest poszukiwanie przestrzeni modelu w celu optymalizacji jego parametrów.

8.1 Selekcja cech

Problem selekcji cech jest niezwykle szerokim polem nauki, cieszącym się w ostatnich latach coraz większym zainteresowaniem. Dowodem tego może być gwałtownie wzrastająca liczba publikacji związanych z tym tematem. Pociąga to jednocześnie coraz większą liczbę możliwych sposobów rozwiązania problemu doboru optymalnego podzbioru cech.
Algorytmy metod selekcji cech można podzielić na różne sposoby, jednak najbardziej istotnym kryterium podziału wydaje się być podział na metody filtrów (ang. filter methods), metody opakowujące (ang. wrappers methods)[96, 95], oraz metody wbudowane (ang. embedded methods).
Pierwsza z grup metod przeprowadza niezależną od procesu klasyfikacji selekcję cech. Z reguły, jako kryterium działania filtrów używa się różnych statystyk, które oceniają

cechy pod względem ich korelacji z etykietami oraz (lub) usuwają ze zbioru cechy redundantne (częstym kryterium filtracji cech jest optymalność w sensie Bayesowskim). Druga zaś grupa metod - opakowujące - do oceny jakości cech używają klasyfikatora i wybierają z całego zbioru atrybutów jedynie te, które pozwalają na najlepszą jego jakość klasyfikacji. Podejście to ma bardzo silne podstawy teoretyczne, co zostało pokazane w pracach Kohaviego, gdyż zbiór optymalnych cech w sensie Bayesowskim nie oznacza optymalnego zbioru cech dla danego klasyfikatora. Ostatnią grupą metod są algorytmy, które posiadają wbudowaną właściwość selekcji cech, ich typowym przykładem są algorytmy drzew decyzji. Osobną możliwą grupą są metody łączące w sobie różne przedstawione powyżej podejścia. Charakteryzują się one tym, iż wstępnej oceny dokonuje inny klasyfikator niż ostateczny lub gdy cząstkowe wyniki filtrów są następnie oceniane przez klasyfikator końcowy. Czasami do oceny istotności cechy używana jest dokładność klasyfikacji w zamian za kryteria statystyczne bądź, w przypadku bardzo dużych zbiorów danych, metody opakowujące poprzedzane są wstępną filtracją cech.
Powszechnie używane są dwa niezależne podejścia we wspomnianych grupach metod - algorytmy rankingowe oraz algorytmy przeszukiwania. Pierwsze dokonują niezależnej oceny każdej z cech indywidualnie, a następnie w oparciu o posortowane wartości istotności dokonywany jest wybór odpowiedniego podzbioru cech. Natomiast metody przeszukiwania korzystają z różnych algorytmów przeszukiwania grafów (drzew) w celu optymalizacji zbioru atrybutów. Drugie z wymienionych podejść charakteryzuje się dużo większą dokładnością, a uzyskiwane rezultaty z reguły przewyższają wyniki metod rankingowych, choć wymagają one dużego nakładu obliczeniowego. Natomiast metody rankingowe są znacznie szybsze, co jest ich istotną zaletą, jednak nie gwarantują one tak dobrych rezultatów jak metody przeszukiwania.
Ponieważ w systemach reguł-P istotną cechą jest prostota uzyskanego rozwiązania przy jednoczesnej maksymalnej zdolności uogólniania, dlatego w rozprawie skoncentrowano się na analizie metod opakowujących oraz metod hybrydowych łączących metody rankingowe oraz metody opakowujące.

8.1.1 Metody rankingowe

Metody rankingowe są najszybszymi metodami selekcji cech. Bazują one na wyznaczeniu współczynnika $J(\cdot)$ określającego zdolność dyskryminacji lub zależność pomiędzy cechą f a klasą C. Współczynnik ten wyznaczany jest indywidualnie dla każdego z atrybutów, a następnie w zależności od wielkości współczynnika $J(\cdot)$ następuje sortowanie od najbardziej do najmniej istotnej cechy. W ostatnim kroku według określonego kryterium np. zdolności dyskryminacji (funkcja oceń$(\cdot)$), pierwszych n' najlepszych cech wybieranych jest jako podzbiór finalny. Algorytm rankingowy przedstawiony jest na schemacie (8.1.1).

Metody rankingowe używają różnych metod oceny jakości pojedynczej cechy, a jedną z nich jest wykorzystanie algorytmów klasyfikacji do oceny każdej z cech według zależności $J_i = Acc(Model(\mathbf{T}_i, \alpha))$, gdzie α to zbiór parametrów modelu, a Acc to jego dokładność. Innymi powszechnie używanymi kryteriami są miary statystyczne oraz bazujące na teorii informacji. Przykładem takich miar jest metryka, zwana znormalizowanym zyskiem informacji (ang. normalized information gain), znana również jako asymetryczny współczynnik zależności (ang. asymmetric dependency

Schemat 14 Selekcja cech w oparciu o ranking cech

Require: $\mathbf{f}$ {wejściowy zbiór cech}
Require: $J(\cdot)$ {funkcja rankingowa}
  for $i = 1 \ldots n$ **do**
    $a_i \leftarrow J(f_i)$ {oceń i-tą cechę}
  end for
  $\mathbf{f} \leftarrow \text{sort}(\mathbf{f}, \mathbf{a})$ {sortuj cechy wg. istotności}
  $acc \leftarrow 0$
  $\mathbf{f}^a \leftarrow \emptyset$
  for $j = 1 \ldots n$ **do**
    $\mathbf{f}^a = \mathbf{f}^a \cup f_i$ {dodaj do podzbioru $\mathbf{f}^a$ nową cechę}
    $tacc \leftarrow$oceń($\mathbf{f}^a$) {oceń podzbiór cech $\mathbf{f}^a$ }
    if $tacc > acc$ **then**
      $acc \leftarrow tacc$ {jeżeli nowy podzbiór jest lepszy od poprzedniego }
      $\mathbf{f}' \leftarrow \mathbf{f}^a$ {zapamiętaj wynik aktualny podzbiór}
    end if
  end for
  return $\mathbf{f}'$

coefficient, ADC) opisana równaniem (8.1) [158]:

$$ADC(C, f) = \frac{MI\,(C, f)}{H\,(C)} \tag{8.1}$$

gdzie $H(c)$ oraz $H(f)$ to odpowiednio entropia klas i cechy, zaś $MI(C, f)$ to informacja wzajemna (ang. mutual information) pomiędzy klasami C i cechą f określona przez Shanonna [156] jako (8.2)

$$\begin{aligned} H(C) &= -\textstyle\sum_{i=1}^{c} p(C_i) \lg_2 p(C_i) \\ H(f) &= -\textstyle\sum_{x} p(f = x) \lg_2 p(f = x) \\ MI(C, f) &= -I(C, f) + H(C) + H(f) \end{aligned} \tag{8.2}$$

W zależności tej suma po x jest realizowana jedynie dla cech f przyjmujących dyskretne wartości, natomiast cechy ciągłe wymagają zamiany sumy na całkę lub wstępnej dyskretyzacji by możliwa była estymacja prawdopodobieństwa $p(f = x)$.
Innym przykładem metryki zaproponowanym przez Setiono [153] jest miara znormalizowanego względnego zysku informacyjnego (ang. normalized gain ratio) opisanego zależnością (8.3).

$$U_S(C, f) \;=\; \frac{MI(C, f)}{H(f)} \tag{8.3}$$

Inną możliwą normalizacją służącą do wyznaczenia zysku informacyjnego może być miara wyznaczająca entropię klasa-cecha poprzez (8.4).

$$U_H(C, f) = \frac{MI(C, f)}{H(f, C)} \tag{8.4}$$

gdzie $H(f, C)$ jest połączoną entropią zmiennych cecha f klasa C.

Mantaras [113] zaproponował interesujące kryterium D_{ML} spełniające wszystkie aksjomaty odległości, określoną jako (8.5).

$$D_{ML}(f_i, C) \;=\; H(f_i|C) \;+\; H(C|f_i) \tag{8.5}$$

gdzie $H(f_i|C)$ i $H(C|f_i)$ są entropią warunkową zdefiniowaną przez Shanonna [156] jako $H(X|Y) = H(X,Y) - H(Y)$.

Index ważonej połączonej entropii został zaproponowany przez Chi [26] jako (8.6)

$$Ch(f) = -\sum_{k=1}^{N} p(f = x_k) \sum_{i=1}^{K} p(f = x_k, C_i) \lg_2 p(f = x_k, C_i) \tag{8.6}$$

Pewną alternatywą do przedstawionych tu miar teorioinformacyjnych jest statystyka χ^2, która mierzy zależność pomiędzy dwoma zmiennymi losowymi - tu zdefiniowanymi jako relacja pomiędzy cechą a klasą. Statystyka χ^2 zdefiniowana jest wyrażeniem (8.7).

$$\chi^2(f, C) = \sum_{ij} \frac{\left(p(f = x_j, C_i) - p(f = x_j) * p(C_i)\right)^2}{p(f = x_j) p(C_i)} \tag{8.7}$$

gdzie $p(\cdot)$ są odpowiednimi prawdopodobieństwami. Duże wartości χ^2 odpowiadają silnej korelacji pomiędzy cechą a klasą, dzięki czemu możliwe jest wykorzystanie tej statystyki do przeprowadzenia procesu rankingu.

Przedstawione powyżej współczynniki stanowią jedynie wybrane miary mierzące relacje podobieństwa rozkładów klasa - cecha. Możliwe jest więc przedstawienie szeregu innych współczynników, do których należą różne współczynniki statystyczne, jak Pearsonowski współczynnik korelacji [69]. Porównanie wyżej opisanych metod rankingowych opisano w rozdziale 8.1.3.

Jak już wspomniano wcześniej, metody rankingowe są najszybszymi metodami selekcji cechy, gdyż ocenie poddawane są globalnie pojedyncze atrybuty. Takie rozwiązanie pociąga jednak za sobą poważne negatywne konsekwencje, mianowicie w finalnym zbiorze cech pozostają wszystkie atrybuty redundantne $\underset{i \neq j}{\exists} f_i = f_j$. Związane jest to z faktem, iż warunkiem koniecznym, lecz nie wystarczającym redundancji jest równość współczynników rankingowych. Powoduje to, że cechy redundantne po dokonaniu rankingu występują obok siebie i dodanie nowej kolejnej cechy wynikającej z rankingu nie wpływa na jakość wyników klasyfikacji, a jedynie niepotrzebnie zwiększa liczbę cech opisujących dany problem. Rozwiązaniem tego problemu są metody filtrów poprzedzające optymalizację wyboru liczby cech użytych do klasyfikacji, a usuwające z rankingu cechy redundantne. Przykładem takiego rozwiązania jest algorytm zaproponowany przez Biesiadę i Ducha [11], bazujący na statystyce Kolmogorova-Smirnova porównujący skumulowane rozkłady prawdopodobieństwa dla par cech oceniając czy cechy są redundantne.

8.1.2 Metody przeszukiwania w selekcji cech

Przewagą metod przeszukiwania nad metodami rankingowymi jest większa dokładność uzyskanych wyników. Metody te bazują na algorytmach przeszukiwania, zarówno stochastycznych jak i heurystycznych. Pociąga to jednak za sobą zwiększony koszt obliczeniowy, co w przypadku dużych zbiorów danych (jak przykładowo dostarczonych

podczas zawodów rozgrywanych w ramach konferencji NIPS'2003, gdzie zbiory liczyły odpowiednio 100 tysięcy i 60 tysięcy cech) może być poważnym ograniczeniem. Typowymi rozwiązaniami rodziny algorytmów przeszukiwania są metody selekcji w przód i tył (ang. forward/backward selection) oraz algorytmy genetyczne.

Selekcja w przód. Algorytm selekcja w przód (8.1.2) rozpoczyna od pustego zbioru cech i stopniowo w każdej iteracji dodaje nową cechę, spośród pozostałych cech, która maksymalizuje określone kryterium (z reguły dokładność liczoną w procesie walidacji krzyżowej).

Schemat 15 Selekcja w przód

Require: $\mathbf{f}$
  $\mathbf{f}' \leftarrow \emptyset$
  $n \leftarrow numof(\mathbf{f})$
  repeat
    $chk \leftarrow$ **true**
    for $i = 1 \ldots n$ **do**
      $tacc \leftarrow$oceń$(\mathbf{f}' \cup f_i)$
      if $tacc > acc$ **then**
        $acc \leftarrow tacc$
        $j \leftarrow i$
        $chk \leftarrow$ **false**
      end if
    end for
    if not chk **then**
      $\mathbf{f}' \leftarrow \mathbf{f}' \cup f_i$
      $\mathbf{f} \leftarrow \mathbf{f}$ f_i
    end if
    $n \leftarrow numof(\mathbf{f})$
  until $chk \vee n = 0$
  return $\mathbf{f}'$

Selekcja w tył. Algorytm selekcji w tył (8.1.2) w odróżnieniu od selekcji w przód, rozpoczyna od pełnego zbioru cech i iteracyjnie usuwa te cechy, które psują jakość klasyfikacji. W każdej iteracji usuwana jest jedna cecha mająca najbardziej negatywny wpływ na dokładność klasyfikacji, aż do momentu gdy określone kryterium przestanie rosnąć.

Inne Spośród innych rozwiązań na uwagę zasługują metody uczenia typu *dodaj i odejmij j* [142], gdzie w każdej iteracji dodawanych jest i nowych cech w procesie selekcji w przód, a odejmowanych jest j cech gdzie $i > j$. Metoda ta dokonuje dużo dokładniejszego przeszukania niż proste metody selekcji w przód lub tył, jednak wielokrotnie wzrasta złożoność obliczeniowa takiego rozwiązania. Kolejnym powszechnie stosowanym rozwiązaniem są systemy stochastycznego przeszukiwania jak algorytmy genetyczne [159, 102, 132] lub inne bazujące na systemach immunologicznych [6]. Bardzo dokładne studium procesu selekcji cech z opisem różnych metod i strategii można znaleźć w [68] oraz w repozytorium prowadzonym przez J. Biesiadę w ramach projektu Fsel++

Schemat 16 Selekcja w tył

Require: $\mathbf{f}$
  $\mathbf{f}^a \leftarrow \mathbf{f}$
  for $j = 1 \ldots n$ **do**
    $\mathbf{f}' \leftarrow \mathbf{f}^a$
    for $i = 1 \ldots n$ **do**
      $\mathbf{f}^b \leftarrow \mathbf{f}' \setminus f_i$
      $tacc \leftarrow$oceń$(\mathbf{f}^b)$
      if $tacc > acc$ **then**
        $acc \leftarrow tacc$
        $\mathbf{f}^a \leftarrow \mathbf{f}^b$
      end if
    end for
  end for
  return $\mathbf{f}'$

[12], gdzie zgromadzonych zostało ponad 724 różnych publikacji wyłącznie o tematyce selekcji cech.

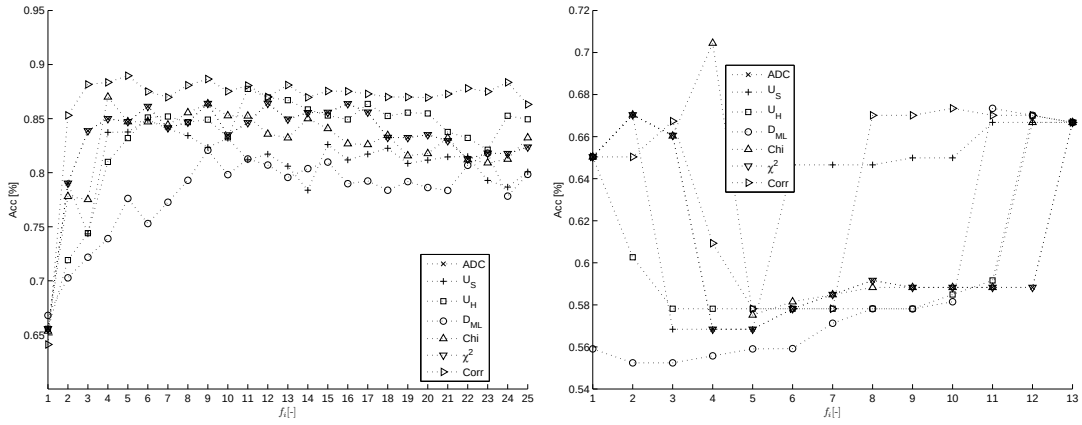
8.1.3 Analiza skuteczności algorytmów selekcji cech

Duża liczba różnych algorytmów selekcji cech wymaga przeprowadzenia analizy w celu wyznaczenia tych metod, które najlepiej realizują przedstawione im zadania. W tym celu dokonano empirycznego porównania metod selekcji zarówno w oparciu o ranking, jak i metody przeszukiwania. Weryfikacji poszczególnych metod dokonano porównując wyniki klasyfikacji na zbiorach opisanych w rozdziale 12. Procedura testowa polegała na wybraniu przez algorytm selekcji cech podzbioru atrybutów i na tym podzbiorze przeprowadzenie testu walidacji krzyżowej klasyfikatora. Dodatkowo metody rankingowe przetestowano na sztucznych zbiorach danych o znanym rozkładzie klas i istotności poszczególnych cech (zbiory *Gauss1* oraz *Gauss2*). Jako klasyfikatory zostały wykorzystane zaproponowane w pracy algorytmy ekstrakcji reguł prototypowych typu k-NN- algorytm CFCM w połączeniu z LVQ oraz optymalizacją liczby prototypów poprzez algorytm wyścigu, jak również algorytm OPTDL. Do wyznaczenia wskaźników (8.1) -(8.7) wykorzystano pakiet selekcji cech Fsel++ [12]

Wyniki

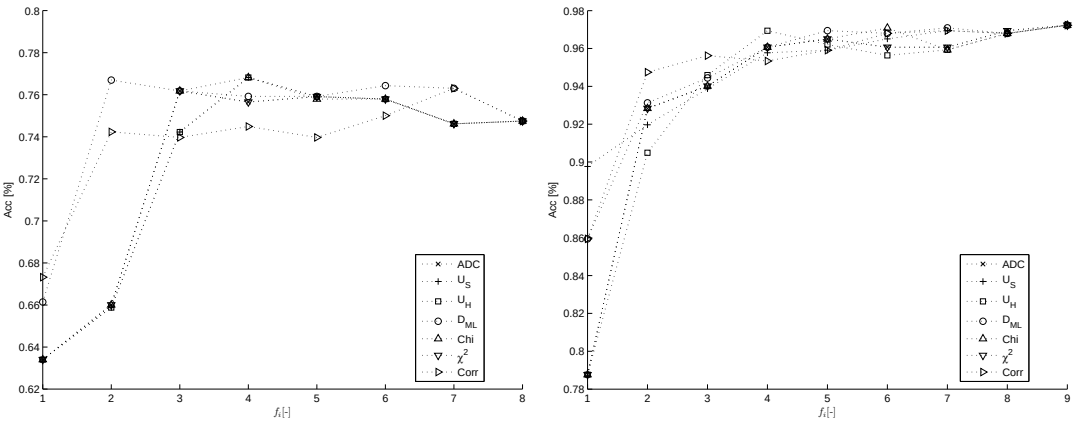
Metody rankingowe. Wstępnie każdy z przedstawionych w pracy współczynników rankingowych poddano ocenie na znanych zbiorach nazwanych *Gauss1* oraz *Gauss2*. Zbiory te są sztucznymi zbiorami danych o odpowiednio 4 i 8 cechach ciągłych. Obydwa zbiory składają się z 4 klas i 4000 wektorów, po 1000 wektorów przypadających na każdą z klas. Zbiór *Gauss1* stanowi zbiór czterech Gaussowskich skupisk w kolejnych wymiarach coraz bardziej nachodzących na siebie. Centrum pierwszego klastra położone jest we współrzędnych $\mathbf{p}_1 = [0, 0, 0, 0]^T$, a kolejnych $\mathbf{p}_2 = a \cdot [1, 1/2, 1/3, 1/4]^T$, $\mathbf{p}_3 = 2a \cdot [1, 1/2, 1/3, 1/4]^T$ oraz $\mathbf{p}_4 = 3a \cdot [1, 1/2, 1/3, 1/4]^T$. Taki układ klastrów powoduje, że współczynniki istotności cech powinny przedstawiać się następująco $[f_1 > f_2 > f_3 > f_4]$. Zbiór *Gauss2* jest rozszerzeniem zbioru *Gauss1* o dodatkowe 4 liniowo zależne cechy,

według zależności $f_i{+}4 = 2f_i{+}\epsilon$ gdzie ϵ stanowi dodany szum o rozkładzie jednostajnym. Wyniki obliczeń przeprowadzone na danych wygenerowanych sztucznie wykazały równoważność wszystkich metryk, które jednogłośnie i bezbłędnie wskazały na istotność poszczególnych cech. Jednak wyniki uzyskane dla realnych zbiorów danych wykazują zmiany porządku hierarchii istotności cech. Różnice te były możliwe do zaobserwowania również ze względu na jakość dyskretyzacji, gdyż do estymacji $p(f = X_j, C_i)$ posłużono się histogramami o 5, 10 oraz 15 przedziałach. Dlatego też do porównania współczynników wykorzystano wartości rankingowe, które dla danej dyskretyzacji najlepiej wypadły w ocenie dokładności klasyfikacji. Proces oceny jakości współczynników rankingowych został podzielony na dwa etapy, gdzie w pierwszym dokonano rankingu cech, a następnie w oparciu o uzyskane wyniki przeprowadzono testy dziesięciokrotnej walidacji krzyżowej klasyfikatora.
Wyniki obliczeń przedstawiono na wykresach rys.(8.1) dla algorytmu CFCM+LVQ oraz na wykresach rys.(8.2) dla algorytmu OPTDL. Poszczególne wykresy przedstawiają zależność dokładności klasyfikacji w funkcji liczby n-pierwszych cech o najwyższej wartości współczynnika dla 4 różnych losowo wybranych zbiorów danych.



(a) Zbiór *jonosfera*

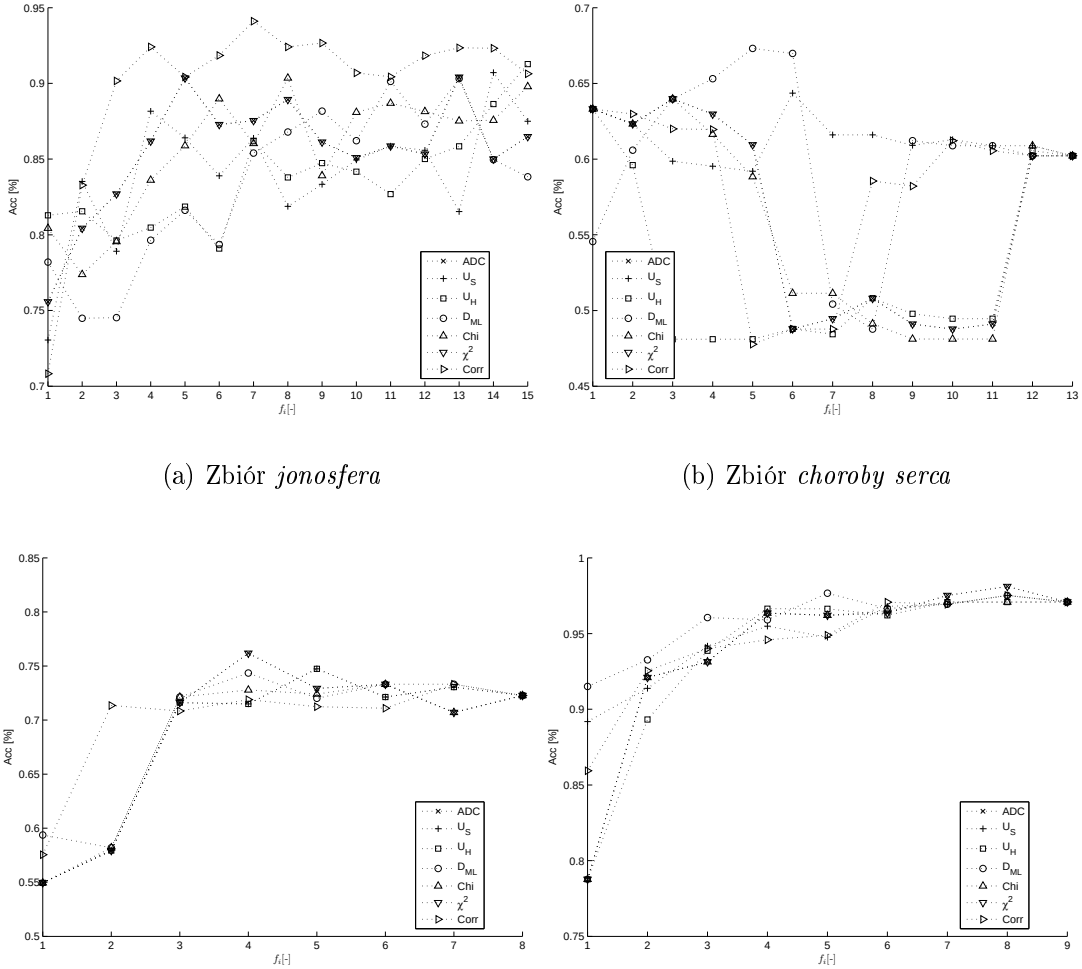
(b) Zbiór *choroby serca*



(c) Zbiór *cukrzyc*

(d) Zbiór *rak piersi*

Rysunek 8.1: Wyniki rankingu dla algorytmu CFCM+LVQ



(a) Zbiór *jonosfera*

(b) Zbiór *choroby serca*

(c) Zbiór *cukrzyc*

(d) Zbiór *rak piersi*

Rysunek 8.2: Wyniki rankingu dla algorytmu OPTDL

Zbiorcze porównanie. Podobnie jak w przypadku oceny metod rankingowych posłużono się tutaj dwuetapowym procesem testowania. W części pierwszej dokonano selekcji cech w oparciu o algorytmy *selekcji w przód*, *selekcji w tył* stosując metodę opakowywania klasyfikatorem oraz metodę selekcji w oparciu o drzewa decyzji, a następnie w drugim kroku dokonano klasyfikacji w procesie dziesięciokrotnej walidacji krzyżowej. Zebrane wyniki dla algorytmu klasyfikacji CFCM+LVQ przedstawiono w tabeli (8.1.3), gdzie dodatkowo umieszczono informację o liczbie wybranych prototypów (l) oraz liczbie cech (f), przy czym dla każdej z metod optymalizowana była jedynie dokładność klasyfikacji. Podobnego porównania dokonano również dla algorytmu OPTDL. W typ przypadku ze względu na złożoność obliczeniową algorytmów selekcji oraz klasyfikatora dokonano porównania metod selekcji bazujących na metodzie rankingowej, gdzie do oceny jakości cech posłużono się współczynnikiem korelacji oraz metody selekcji bazującej na drzewach decyzji. W porównaniu zastosowano dwie metody rankingowe typu *selekcja w przód* oraz *selekcja w tył* gdzie kolejne cechy dodawane są wg. wartości współczynnika rankingowego, aż do osiągnięcia pierwszego maksimum. Taka procedura dodatkowo ogranicza złożoność obliczeniową, jednakże powoduje, że osiągane

	Selekcja w tył			Selekcja w przód			Selekcja drzewem		
Zbiór	Dokładność	f	l	Dokładność	f	l	Dokładność	f	l
Wyrostek rob.	85.70 ± 13.98	6	2	84.86 ± 14.38	4	2	84.86 ± 15.85	2	2
Winorośl	97.25 ± 2.90	11	3	95.06 ± 4.79	5	7	92.57 ± 6.05	6	3
Cukrzyca	77.22 ± 4.36	7	3	76.96 ± 3.01	4	3	75.65 ± 2.43	2	4
Jonosfera	87.81 ± 6.90	32	6	92.10 ± 4.25	5	6	87.57 ± 7.11	2	4
Choroby serca	85.49 ± 5.59	7	3	84.80 ± 5.23	3	2	84.80 ± 5.23	3	2
Irysy	95.33 ± 6.32	3	5	97.33 ± 3.44	2	4	97.33 ± 3.44	2	4
Sonar	66.81 ± 20.66	56	4	75.62 ± 12.97	4	3	74.62 ± 12.40	1	2
Choroby wątroby	67.24 ± 8.79	5	2	68.45 ± 6.40	4	2	66.71 ± 9.05	2	2
Rak piersi	97.82 ± 1.96	7	3	97.81 ± 2.19	5	3	97.23 ± 2.51	5	4
Lancet	95.66 ± 2.36	7	4	94.65 ± 2.64	5	4	93.33 ± 3.20	4	3

Tablica 8.1: Porównanie algorytmów selekcji cech dla algorytmu CFCM+LVQ

	Ranking w tył			Ranking w przód			Selekcja drzewem		
Zbiór	Dokładność	f	l	Dokładność	f	l	Dokładność	f	l
Wyrostek rob.	87.77 ± 9.68	6	1	87.77 ± 9.68	6	1	81.88 ± 17.58	2	1
Winorośl	96.70 ± 4.67	13	5	75.34 ± 11.45	1	3	89.91 ± 8.47	4	5
Cukrzyca	76.43 ± 4.66	3	4	76.43 ± 4.66	3	4	72.79 ± 5.43	4	1
Jonosfera	93.79 ± 4.30	27	5	93.79 ± 4.30	27	5	88.68 ± 4.83	2	1
Choroby serca	84.15 ± 5.58	3	3	84.15 ± 5.58	3	3	84.15 ± 5.58	3	3
Irysy	98.00 ± 3.22	2	2	98.00 ± 3.22	2	2	98.00 ± 3.22	2	2
Sonar	78.55 ± 15.71	13	3	78.55 ± 15.71	13	3	73.17 ± 11.33	1	1
Choroby wątroby	59.36 ± 9.40	6	5	56.52 ± 9.06	1	1	64.06 ± 4.96	2	4
Rak piersi	97.23 ± 2.10	9	1	91.96 ± 4.72	1	1	96.35 ± 2.48	5	1
Lancet	94.35 ± 2.94	7	3	94.35 ± 2.94	7	3	94.35 ± 1.63	4	3

Tablica 8.2: Porównanie algorytmów selekcji cech dla algorytmu OPTDL

są maksima lokalne. W tabeli tab.(8.1.3) użyto identycznych oznaczeń jak w tab.(8.1.3).

Podsumowanie wyników Przedstawione wyniki wskazują, iż dobór odpowiedniej metody selekcji cech jest niezmiernie istotny. Wyniki badań opublikowane w [68] oraz rezultaty konkursów selekcji cech [128] sugerują, że nie istnieje istotny związek pomiędzy typem klasyfikatora a odpowiednią metodą rankingową selekcji cech. Dlatego też podczas wyboru wskaźnika rankingowego można się kierować prostotą jego wyznaczenia (wskaźniki teorioinformacyjne wymagają wstępnej dyskretyzacji) lub powinien być realizowany w procesie meta uczenia.
W przypadku reguł prototypowych dają się jednak zaobserwować pewne tendencje wynikające z konieczności znalezienia kompromisu pomiędzy dokładnością a złożonością modelu. Analiza załączonych wyników wskazuje, iż metody *selekcji w przód* oraz *selekcji w tył* charakteryzują się największą dokładnością i przewyższa pozostałe metody. Należy jednak zwrócić uwagę na złożoność obliczeniową tych metod, jest ona rzędu $!!!!!XXXXXXX!!!!!$ - powoduje to, że wykorzystanie ich w realnych problemach analizy danych jest ograniczone ze względu na koszt obliczeń. Ważnym podkreślenia jest również problem osiągania przez opisane metody ekstremów lokalnych (pierwszego napotkanego maksimum). Widać to porównując wyniki uzyskane dla zbioru *choroby wątroby* i algorytmu selekcji bazującego na drzewach decyzji, gdzie ta metoda selekcji dała lepsze rezultaty oraz mniejszą liczbę ostatecznie wybranych cech. Możliwością

obejścia problemu ekstremów lokalnych są algorytmy przeszukiwania typu *dodaj i odejmij j*, które jednak dodatkowo zwiększają złożoność obliczeniową.
Wady tej pozbawione są metody rankingowe, których zaletą jest niska złożoność obliczeniowa odpowiadająca liniowej zależności w funkcji liczby cech ($O(|\mathbf{f}|)$). Jednak metody te mogą być niestabilne w funkcji uzyskiwanej dokładności - obrazują to wykresy rys.(8.1(a)) i rys.(8.1(b)) dla algorytmu CFCM+LVQ oraz rys.(8.2(a)) i rys.(8.1(b)) dla algorytmu OPTDL. Uzyskane wyniki dla metod rankingowych nie pozwalają na jednoznaczne wyłonienie najlepszego wskaźnika, dlatego też propozycją rozwiązania są komitety metod rankingowych, w których o wadze danej cechy decyduje częstość jej występowania na określonej pozycji. Na uwagę zasługuje również duża skuteczność wskaźnika rankingowego, jakim jest współczynnik korelacji. Wśród jego zalet należy zaznaczyć niski koszt jego wyznaczenia oraz niezależność od problemu estymacji prawdopodobieństwa.
Rozwiązaniem kompromisowym wydaje się być selekcja cech wykorzystująca metodę drzew decyzji. Cechuje się ona umiarkowaną złożonością obliczeniową $!!!!!!XXXXXO(|\mathbf{f}|\log|\mathbf{f}|)!!!!!$, a jednocześnie w miarę dobrymi rezultatami selekcji.
Metodą łączącą zalety algorytmów drzew decyzji oraz algorytmów przeszukiwania jest łączenie różnych metody selekcji. Przykładowym rozwiązaniem jest tutaj dwuetapowy proces selekcji, gdzie w pierwszym kroku wybierany jest początkowy podzbiór cech, realizowany poprzez metodę selekcji w oparciu o drzewa decyzji, a następnie, w drugim kroku selekcja w przód dodająca do wybranego podzbioru cechy maksymalizujące dokładność lub też wybór cech na podstawie drzewa nie przyciętego (pełnego) a następnie usuwanie nieistotnych cech zgodnie z algorytmem selekcji w tył. Algorytmy te wymagają jednak przeprowadzenia dalszych badań.

8.2 Meta uczenie

Meta uczenie można podzielić na dwa niezależne problemy - selekcję optymalnych parametrów modelu oraz selekcję optymalnego algorytmu uczenia. W pierwszy przypadku dążymy do dobrania optymalnych hiperparametrów modelu, które maksymalizują daną funkcję kryterialną - takimi parametrami mogą być współczynnik uczenia czy też rodzaj funkcji odległości. Problem selekcji modelu to problem doboru najlepszego algorytmu uczenia (np. problem doboru najlepszej metody selekcji prototypów lub typu reguł-P) oraz odpowiednich transformacji danych.
W szczególności meta uczenie obejmuje również zagadnienie oceny jakości modelu, strategię poszukiwania optymalnego algorytmu oraz problem redukcji złożoności obliczeniowej.

8.2.1 Metody oceny modelu

Szacowanie jakości modelu

Jednym z istotnych problemów jest ocena jakości zbudowanego modelu. Ogólnie istnieje wiele podejść do tego zagadnienia m.in. przedstawione w [61]. Najbardziej trywialnym jest testowanie modelu bazujące na dwóch różnych zbiorach - treningowym, używanym do uczenia modelu i testowym, oceniającym jakości modelu. Takie rozwiązanie ma wiele wad. Jego poważnym ograniczeniem jest zależność uzyskanych wyników od podziału na część treningową i testową, co w znaczny sposób zafałszowuje prawdziwe możliwości

modelu. Jedyną możliwością w miarę rzetelnej oceny przy wykorzystaniu podziału trening/test jest analiza danych o znanym rozkładzie prawdopodobieństwa (np. w celach porównawczych różnych metod) - gdzie możliwe jest stworzenie zbioru testowego o rozmiarze znacznie przekraczającym rozmiar zbioru treningowego ($m_{test} \gg m_{train}$).
Metodą najpowszechniej stosowaną w problematyce oceny jakości modelu jest k-krotna walidacja krzyżowa (ang. k-fold cross validation), występująca w dwóch odmianach. Proces walidacji polega tu na podziale zbioru treningowego $\mathbf{T}$ na k równych części, gdzie system uczony jest k-krotnie na $k-1$ częściach, a testowany na pozostałej jednej części zbioru $\mathbf{T}$. W każdej iteracji następuje zmiana części testującej, tak że w rezultacie system testowany jest na wszystkich wektorach zbioru treningowego. Poszczególne odmiany k-krotnej walidacji krzyżowej różnią się tym, iż w przypadku walidacji stratyfikowanej dąży się do tego, by zapewnić stały rozkład prawdopodobieństwa występowania klas w każdej z k części. Pomiar dokładności klasyfikacji w przypadku walidacji krzyżowej liczony jest jako średnia dokładność dla każdego z k procesów.
Problem który występuje w przypadku walidacji krzyżowej to dobór odpowiedniej wartości k. Mała wartość k pozwala na bardziej rzetelną ocenę pojedynczej walidacji, gdyż stosunek części walidującej do uczącej jest stosunkowo duży. Rozwiązanie takie zmniejsza jednak ilość powtórzeń procesu uczenia, co negatywnie wpływa na jakość oceny wartości średniej. Należy również zauważyć kolejny problem, jakim jest reprezentatywność danych uczących. Małe wartości k mogą doprowadzić do sytuacji, gdzie pewne drobne, acz istotne obszary będą mało licznie reprezentowane w części uczącej, co może doprowadzić do ich pominięcia. Problem ten nie występuje dla dużych wartości k, gdzie dodatkowo poprawia się jakość oceny wartości średniej. Nie należy jednak zapominać o wadzie metody, jaką jest wzrost złożoności obliczeniowej, co może w znacznym stopniu ograniczać praktyczne aplikacje (najczęściej przyjmuje się $k = 10$).
Skrajnym przypadkiem walidacji krzyżowej jest test *jeden pozostaw* (ang. leave one out), gdzie system testowany jest m-krotnie zawsze na pojedynczym wektorze. W literaturze dowodzi się, że test ten jest „niemal" nieobciążony [149], jednakże na uwagę zasługuje fakt, iż nie pozwala on na wyznaczenie wariancji modelu, którą łatwo daje się wyliczyć w przypadku $k \ll n$ jako odchylenie standardowe wyników uzyskanych w poszczególnych walidacjach. Informacja taka może być bardzo istotna, dając dodatkowe wskazówki o zbudowanym modelu.
Rozwinięciem testu walidacji krzyżowej jest test, w którym proces walidacji krzyżowej powtarzany jest wielokrotnie (tak zwany X-test). Dzięki takiemu rozwiązaniu zwiększa się liczba prób pojedynczych ocen, jednocześnie uniezależniając się od losowości podziału na k podzbiorów, co ma szczególne znaczenie dla małych zbiorów danych.
W literaturze spotykane są również inne metody selekcji modelu, jak bazujące na teorii Bayesowskiej [152, 123]. W metodzie tej zakłada się, że parametry modelu takie, jak współczynniki uczenia, stopień metryki L są traktowane - podobnie jak cechy zbioru uczącego - jako zmienne losowe poddając je analizie Bayesowskiej. Jednak przeprowadzone testy empiryczne wskazują, że dla małych zbiorów danych ze względu na problem estymacji prawdopodobieństwa możliwe jest występowanie błędów tej metody.
Bardzo dobrą metodą testowania modelu jest tak zwany *Bootsrtaping* lub test *Monte Carlo* [61], który polega na wielokrotnym losowaniu zarówno części uczącej, jak i testującej ze zwracaniem. Takie rozwiązanie powoduje bardziej naturalny (zbliżony do rzeczywistego) rozkład danych w obydwu zbiorach, jednakże wymagane jest, by proces podziału na część treningową i testową realizowany był wielokrotnie. Niestety determinuje to bardzo dużą złożoność obliczeniową tej metody. Jest ona szczególnie

często stosowana w przypadkach małolicznych zbiorów danych, dla których metody walidacji krzyżowej nie dają zadowalających efektów.

Wybór najlepszego modelu

Omówione powyżej metody weryfikacji jakości modelu pozwalają na estymację jego dokładności, liczoną najczęściej jako średnią z uzyskanych wyników walidacji (dla metod walidacji krzyżowej oraz bootstrap). Jednak ocena i wybór modelu na podstawie jedynie wartości średniej dokładności $\overline{Acc}$ obarczona może być błędem, gdyż tym sposobem nie jest uwzględniania wariancja modelu. Wariancja modelu może być natomiast interpretowana jako wariancja uzyskanych wyników dokładności. Przeprowadzone testy wskazują, iż wpływ wariancji jest bardzo istotny, gdyż średnia dokładność nie uwzględnia rozrzutu wartości dokładności poszczególnych walidacji. Ma to również istotny wpływ na stabilizację wyników szacowania.
Do wyboru najlepszego modelu niezbędne jest więc porównanie zarówno wartości średniej z różnych walidacji, jak i jej odchylenia standardowego. Można w tym celu posłużyć się różnymi testami statystycznymi, jednak najprostszą formą jest analiza dokładności zdefiniowana jako (8.8).

$$\overline{Acc} - std \tag{8.8}$$

Metoda ta pozwala na wybór modelu na podstawie pesymistycznego oszacowania dokładności. Stanowi ona w pewnym sensie uproszczenie testu T-Studenta dla średnich z próby.

8.2.2 Metody przeszukiwania

Podobnie jak w przypadku selekcji prototypów oraz selekcji cech, również w przypadku selekcji modelu największą popularnością cieszą się metody przeszukiwania. Jednakże z uwagi na konieczność optymalizacji wielu parametrów modeli selekcji cech, proces ten staje się bardzo zachłanny obliczeniowo. Główną wadą przestawionego na schemacie (8.2.2) algorytmu jest zagnieżdżenie jednego procesu przeszukiwania w drugim. Przykładem takiego zjawiska może być selekcja cech połączona z selekcją prototypów, gdzie obydwa zadania realizowane są przez proces szukania. Dlatego też konieczne staje się rozważenie algorytmów przyspieszających i upraszczających, redukując również złożoność obliczeniową procesu przeszukiwania.

Strategia 1 Jedną z możliwości poprawy jest wykorzystanie metody filtrów w selekcji cech (np. poprzez użycie drzew decyzji), dzięki czemu procesy selekcji cech i optymalizacji parametrów modelu stają się niezależne, znacznie redukując złożoność obliczeniową w porównaniu np. do metod opakowujących.

Strategia 2 Inną możliwością jest rozpoczęcie procesu uczenia od optymalizacji parametrów modelu realizowanego na całym zbiorze danych lub po uprzedniej selekcji cech zrealizowanej w oparciu o metody filtrów (strategia 1). W kolejnym kroku następuje proces selekcji cech metodami opakowującymi, algorytmem selekcji w tył, usuwając jedynie te cechy, które psują dokładność danego modelu.

Schemat 17 Przykładowy algorytm meta-uczenia

Require: **T**,
Require: **M** {Zbiór klasyfikatorów}
Require: **S** {Zbiór metod selekcji cech}
  $k \leftarrow$ sizeof(**M**)
  $t \leftarrow$ sizeof(**S**)
  $acc \leftarrow 0$
  for j=1...k **do**
    for $i = 1 \ldots t$ **do**
      $tacc \leftarrow$ Validate($[S_i, M_j]$, **T**)
      if $tacc > acc$ **then**
        $acc \leftarrow tacc$
        $S' \leftarrow S_i$
        $M' \leftarrow M_j$
      end if
    end for
  end for
  return M', S'

Strategia 3 Ostatnie proponowane rozwiązanie wykorzystuje możliwości algorytmów uczenia, bazujących na metodach optymalizacji gradientowej posiadających zdolność douczania. W przypadku reguł prototypowych dotyczy to algorytmu LVQ, który pozwala na rozszerzenie strategii 2. W metodzie tej, po każdym usunięciu cechy następuje douczenie systemu poprzez kilka iteracji algorytmu LVQ1 bądź LVQ2.x. Dzięki takiemu rozwiązaniu znacznie skraca się czas uczenia modelu, co sprzyja dalszej optymalizacji. Ponadto douczanie modelu bazuje na wykorzystaniu uprzednio zdobytej wiedzy, co wpływa na jakość uzyskanych rezultatów pomijając problem inicjalizacji.

8.2.3 Redukcja złożoności obliczeniowej w metodach walidacji krzyżowej oraz Bootstrap

Bardzo ciekawą metodę redukcji złożoności obliczeniowej zaproponowali w swojej pracy Maron and Moore [116]. W swych badaniach analizowali oni możliwość skrócenia procesu obliczeniowego poprzez zaprzestanie walidacji modelu, który jest znacząco gorszy od aktualnie najlepszego. Ich algorytm wykorzystywał statystykę T-Studenta do oceny wyników w kolejnych krokach k-krotnej walidacji krzyżowej. Jeżeli aktualnie analizowany model po s krokach, gdzie $s < k$, wykazywał istotnie gorsze wyniki niż najlepszy z dotychczasowych modeli, proces walidacji zostawał wstrzymany, a aktualny model wycofany jako bezużyteczny.

Rozdział 9

Relacje pomiędzy regułami rozmytymi a prototypowymi

9.1 Wstęp

Do budowy rozmytych klasyfikatorów regułowych najczęściej używa się modelu Takagi-Sugeno-Kanga (TSK), gdzie Cordon [31] definiuje trzy podstawowe typy reguł, a dokładniej trzy typy konkluzji:

1. jeżeli ... to C($\mathbf{x}$)=C_i, $z = 1$ - reguła ma bezpośrednio przypisaną jedną wartość etykiety ze współczynnikiem spełnienia z_i reguły równym 1

2. jeżeli ... to C($\mathbf{x}$)=C_i,$z = [0, 1]$ - konkluzja reguły wskazuje na pojedynczą klasę z odpowiednim współczynnikiem wsparcia (spełnienia) konkluzji należącym do przedziału $[0, 1]$

3. jeżeli ... to C($\mathbf{x}$)=$C_{1...c}$, $z_{1...c} = [0, 1]$ - konkluzja reguły definiuje współczynniki wsparcia (spełnienia) konkluzji odpowiednio dla każdej z klas

przy czym każda konkluzja składa się z pary $< C_i, z_i >$, gdzie C_i jest singletonem definiującym konkluzję, natomiast z_i jest współczynnikiem określającym jej pewność. Model ten spełnia również właściwości modelu Mamdaniego, przy założeniu singletonów w konkluzji. Kuncheva rozszerza tą grupę o konkluzję, gdzie dla każdej z klas w ramach pojedynczej reguły definiuje się wartości lingwistyczne, dzięki czemu zastosować można pełny model wnioskowania Mamdaniego.
Zarówno analiza funkcjonalna, jak i analiza interpretacji systemów rozmytych wskazują na równoważność regułowych klasyfikatorów rozmytych oraz metod bazujących na wektorach referencyjnych. Dowodzi to między innymi praca Bilgica oraz Turksena [13], gdzie autorzy podają pięć metod interpretacji funkcji przynależności:

1. jako wartość prawdopodobieństwa (ang. likelihood view)

2. jako wartości zbiorów losowych (ang. random set view)

3. jako stopień podobieństwa (ang. similarity view)

4. jako miara funkcjonalności (ang. utility view)

5. jako wartość wynikająca z teorii pomiarów (ang. measurement view)

Na uwagę zasługuje tu interpretacja stopnia przynależności jako podobieństwa.
Pierwsze bezpośrednie przykłady literaturowe o równoważności funkcjonalnej systemów rozmytych i prototypowych były opisywane przez Janga [79] i dotyczyły sieci RBF (równoważne z rozmytym modelem TSK typu 2). W [100] Kuncheva pokazała równoważność systemów rozmytych oraz metod statystycznych bazujących na funkcjach jądrowych takich jak kNN $\equiv$ TSK typ 1 oraz klasyfikator Parzena $\equiv$ TSK typ 2. Ostatecznie zaproponowała bardzo ogólny model bazujący na prototypach, powstały na styku z systemami rozmytymi nazwany Uogólniony Klasyfikator Najbliższego Prototypu (GNPC) [103], omówiony również w rozdziale 4. W książce [101] ta sama autorka pokazuje równoważność GNPC z omówionymi modelami TSK. Głosi również tezę o równoważności klasyfikatorów rozmytych oraz prototypowych, wskazując, iż każdy system rozmyty można zapisać jako równoważny system prototypowy, czyli jako zbiór reguł prototypowych.

9.2 Różnice pomiędzy systemami reguł prototypowych oraz rozmytych

Równoważność systemów prototypowych oraz rozmytych uzyskuje się przez założenie separowalnych funkcji podobieństwa lub odległości. Separowalne funkcje odległości charakteryzują się możliwością dekompozycji poszczególnych składników odległości na niezależne składowe dla każdego z atrybutów. Przykładowym rozwiązaniem jest tutaj sieć neuronowo-rozmyta FSM [54], która pozwala na interpretację wyników jako zbioru reguł rozmytych. Jednak jej sposób działania opiera się w rzeczywistości na selekcji prototypów i ich interpretacji po uprzedniej dekompozycji jako zbioru reguł rozmytych. Nie wszystkie jednak funkcje podobieństwa lub odległości posiadają właściwość separowalności. Przykładem tego może być odległość cosinusowa, której nie można rozseparować na niezależne składowe atrybutów lub też funkcje bazowe stosowane w przypadku sieci neuronowych typu RBF, za wyjątkiem funkcji Gaussowskiej.
Różnice pomiędzy systemami rozmytymi a prototypowymi uwydatniają się w lingwistycznej interpretacji reguł. Reguły rozmyte opisują dany problem posługując się przymiotnikami wyrażającymi elementy reguł, podczas gdy reguły prototypowe do opisu wykorzystują obiekty (9.1).

$$\text{jeżeli O jest podobny do P to kategoria(O)=kategoria(P)} \tag{9.1}$$

gdzie O to badany obiekt, zaś P to jeden z prototypów - obiektów referencyjnych. Przykładem lingwistycznego wyrażenia reguły prototypowej może być:

jeżeli obiekt A jest najbardziej podobny do koszykarza to A gra w piłkę

gdzie opis koszykarza podany jest niejawnie, ale można go zdekomponować (o ile funkcja podobieństwa na to pozwala) na zbiór przesłanek równoważnych opisowi zbiorów rozmytych.
Taka właściwość stawia reguły prototypowe jako nadrzędne (bardziej ogólne), gdyż operują one na wyższym poziomie abstrakcji niż reguły rozmyte. Rozwiązanie to ma jednak wady, gdyż ceną ogólności może być redundancja informacji zawarta w regułach prototypowych, wynikająca z konieczności reprezentacji prototypu jako wektora $\mathbf{p}$, podczas gdy w przypadku reguł rozmytych istnieje większa możliwość uproszczenia

zapisu przesłanek oraz wykorzystanie rozmytych operatorów negacji.
Poważnym ograniczeniem systemów rozmytych jest ich zdolność analizy danych symbolicznych, dla których trudno jest zdefiniować poprawną funkcję przynależności, a tym bardziej nadać jej interpretację lingwistyczną. Problem ten wraz z propozycją rozwiązania opisano m.in. w [126]. Niemniej jest to poważne zagadnienie, z którym dobrze radzą sobie systemy prototypowe. Należy tu zwrócić uwagę, iż początkowo były one specjalizowane do analizy właśnie danych symbolicznych, jak metody z rodziny CBR [120]. Dużo prac poświęconych problematyce cech symbolicznych było prowadzonych również przez społeczność związaną z uczeniem instancyjnym (ang. instance based learning, IBL), gdzie opracowanych zostało szereg miar z rodziny VDM [162] oraz odpowiednie miary heterogeniczne [181, 180, 16] (rozdział 5).
Istotną zaletą systemów rozmytych jest bogactwo operatorów opracowanych w celu uwypuklenia pewnych właściwości opisu wiedzy. Należy tu wymienić różne operatory iloczynu oraz sumy logicznych przesłanek (ang. T-norm,T-conorm), operatory implikacji oraz inne operatory logiczne. Na rzecz systemów rozmytych przemawia również duża łatwość tworzenia bazy reguł w oparciu o wiedzę eksperta, co stanowiło o sile i popularności systemów reguł rozmytych.

9.3 Odległości a podobieństwo

W metodach bazujących na prototypach możliwe jest wykorzystanie dwóch typów operatorów: odległości ($D(O,P)$) lub podobieństwa ($S(O,P)$)[1]. Pierwsze z wymienionych (odległości) zakładają równoważność obiektów, gdy $D(O,P)=0$, zaś całkowitą różność obiektów, gdy $D(O,P)=\inf$ lub dla odległości znormalizowanych $D'(O,P)=1$. W przypadku podobieństwa równoważność O oraz P określa się po normalizacji jako $S(O,P)=1$, natomiast jej całkowity brak, jako $S(O,P)=0$.
Taka różnorodność powoduje konieczność zdefiniowania operatorów, które pozwolą na wzajemną transformację obydwu typów miar między sobą. Jest to tym bardziej niezbędne, gdyż systemy rozmyte posiadają bezpośrednią interpretację jako metod prototypowych bazujących na podobieństwach. Wzajemne porównanie więc systemów możliwe jest jedynie przy założeniu pokazania równoważności koniunkcji funkcji przynależności z odpowiednimi funkcjami odległości.
Najogólniej wzajemną transformację pomiędzy odległością a podobieństwem można zapisać jako (9.2)

$$\begin{aligned} S(\mathbf{x},\mathbf{p}) &= T\left(D(\mathbf{x},\mathbf{p})\right) \\ D(\mathbf{x},\mathbf{p}) &= T^{-1}\left(D(\mathbf{x},\mathbf{p})\right) \end{aligned} \tag{9.2}$$

gdzie $T(\cdot)$ oznacza monotoniczną funkcję dokonującą transformacji. Najprostszą postacią jest liniowa zależność pomiędzy odległością, a podobieństwem, którą można wyrazić jako (9.3) [70] przy założeniu znormalizowanej odległości.

$$S(O,P) = 1 - D'(O,P) \tag{9.3}$$

Inną możliwą postacią transformacji (9.2) jest funkcja (9.4). Jej cechą jest transformacja addytywnej funkcji odległości do postaci iloczynowej funkcji podobieństwa (9.5).

$$\begin{aligned} S(\mathbf{x},\mathbf{p}) &= \exp\left(-D(\mathbf{x},\mathbf{p})^q\right) \\ D(\mathbf{x},\mathbf{p})^q &= -\ln\left(S(\mathbf{x},\mathbf{p})\right) \end{aligned} \tag{9.4}$$

[1] O oraz P oznaczają tu obiekty, które niekoniecznie posiadają opis wektorowy

$$S(\mathbf{x},\mathbf{p}) = \exp\left(-D(\mathbf{x},\mathbf{p})^q\right) = \exp\left(-\sum_{i=1}^{n} d(x_i,y_i)\right) = \prod_{i=1}^{n} \exp\left(-d(x_i,y_i)\right) \tag{9.5}$$

gdzie α to wykładnik potęgi funkcji odległości Minkowskiego. Innymi możliwościami transformacji odległości do podobieństwa są funkcje (9.6),(9.7) oraz (9.8).

$$S(\mathbf{x},\mathbf{p}) = \frac{1}{1+D^{\alpha}(\mathbf{x},\mathbf{p})} \tag{9.6}$$

$$S(\mathbf{x},\mathbf{p}) = \frac{1}{D^{\alpha}(\mathbf{x},\mathbf{p})} \tag{9.7}$$

$$D^{\alpha}(\mathbf{x},\mathbf{p}) = ctan\left(\frac{\Pi}{2} S\left(x_i,y_i\right)\right) \tag{9.8}$$

Przedstawione metody wzajemnej transformacji odległości i podobieństwa są jedynie przykładowymi przekształceniami [182], których można wyznaczyć znacznie więcej. Zgodnie z definicją Kunchevy użytą w opisie klasyfikatora GNPC [103] odpowiednią funkcją transformującą jest dowolna monotonicznie malejąca funkcja $T(\cdot)$.

9.4 Równoważność odległości oraz funkcji przynależności

Jak wspominano powyżej, z punktu widzenia funkcjonalnego obydwa typy systemów są przy założeniu pewnych ograniczeń równoważne, jednakże każdy z nich ma pewne funkcjonalne zalety. Dla systemów rozmytych jest to bogata liczba różnego rodzaju operatorów, zaś systemy prototypowe cechuje różnorodność funkcji odległości, które jednak muszą spełniać własność separowalności by móc zostać wyrażone w postaci koniunkcji funkcji przynależności. Duże możliwości dla obydwu typów systemów daje więc zbadanie i porównanie funkcji odległości używanych powszechnie w metodach prototypowych oraz funkcji przynależności wraz z odpowiednimi operatorami koniunkcji reguł rozmytych. Dzięki temu uzyskuje się wzrost elastyczności obydwu systemów oraz możliwość wzajemnej transformacji jednych w drugie [42].
W poniższych analizach dla uproszczenia założono model wnioskowania TSK typu (1) co odpowiada prostemu modelowi k-NN.

9.4.1 Od funkcji przynależności do funkcji odległości

Najczęściej stosowanym operatorem przecięcia zbiorów rozmytych jest operator iloczynu algebraicznego, a więc uzyskana przy jego pomocy funkcja podobieństwa przyjmuje postać (9.9)

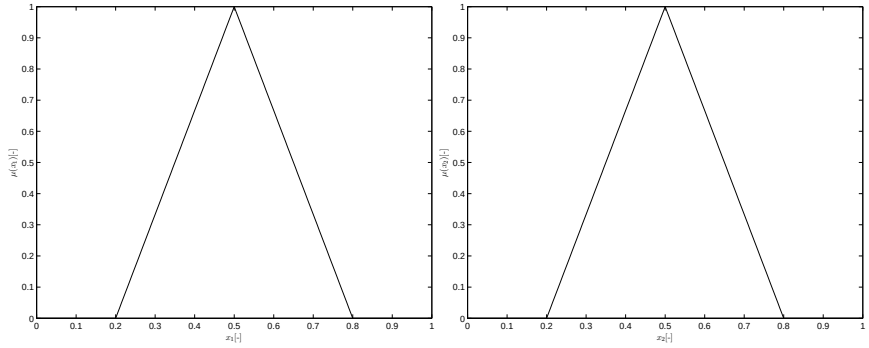
$$S(\mathbf{x},\mathbf{p}) = \prod_{i=1}^{n} (\mu(x_i;p_i)) \tag{9.9}$$

gdzie $\mu(x_i;p_i)$ jest funkcją przynależności, dla której p_i jest parametrem określającym jej położenie. Dla tak zdefiniowanej funkcji podobieństwa (T-norma typu iloczyn) odpowiednia transformacja do postaci odległości realizowana jest poprzez przekształcenie (9.4) [70]. Zastosowanie tej transformacji do iloczynu Gaussowskich funkcji przynależności równoważne jest kwadratowi odległości Euklidesa.

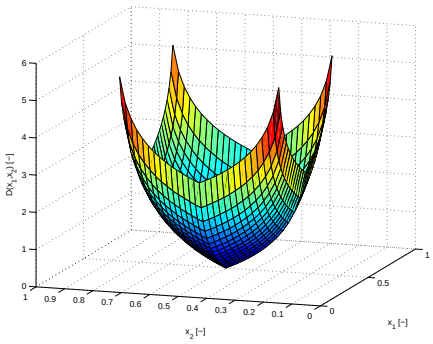
Powyższy przykład stanowi dowód na równoważność systemów TSK typu 1 z Gaussowskimi funkcjami przynależności oraz operatorem T-norma typu iloczyn i klasyfikatora 1-NN.
Identyczna transformacja iloczynu trójkątnych funkcji przynależności prowadzi do logarytmicznej postaci funkcji odległości. Oznacza to, że iloczyn liniowo opadających funkcji przynależności (stożkowa funkcja podobieństwa) (9.10)

$$\mu(x_i; p_i) = \begin{cases} -|x_i - p_i| + 1 & |x_i - p_i| \leq 1 \\ 0 & |x_i - p_i| > 1 \end{cases} \tag{9.10}$$

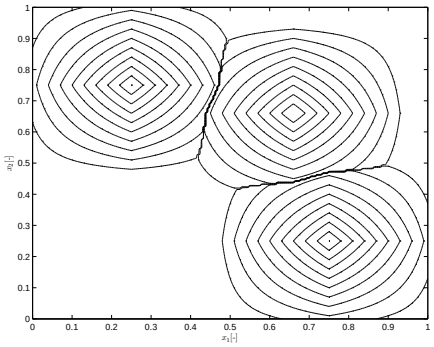
równoważny jest logarytmicznej funkcji odległości. Zostało to pokazane na zależności (9.11) oraz rys.(9.1)



(a) Funkcje przynależności



(b) Funkcja odległości



(c) Przykładowa granica decyzji dla trzech prototypów lub trzech reguł rozmytych

Rysunek 9.1: Trójkątna funkcja przynależności z T-normą iloczynową oraz odpowiadająca jej funkcja odległości.

$$\begin{gathered} D(\mathbf{x}, \mathbf{p}) = -\ln\left(S(\mathbf{x}, \mathbf{p})\right) = \\ = -\ln\left(\prod_{i=1}^{n} \begin{cases} -|x_i - p_i| + 1 & |x_i - p_i| \leq 1 \\ 0 & |x_i - p_i| > 1 \end{cases}\right) \\ D(\mathbf{x}, \mathbf{p}) = \begin{cases} -\sum_{i=1}^{n} \ln(-|x_i - p_i| + 1) & |x_i - p_i| \leq 1 \\ 0 & |x_i - p_i| > 1 \end{cases} \end{gathered} \tag{9.11}$$

Logarytmiczny kształt funkcji odległości wskazuje, iż dla obiektów znajdujących się w pobliżu wektora referencyjnego następuje powolny wzrost odległości, zaś przekroczeniu pewnego progu Θ towarzyszy jej gwałtowny wzrost. Tak zdefiniowana funkcja odległości ma dodatkową własność, którą jest ograniczenie obszaru aktywacji. Oznacza to, że dla skończonej różnicy położenia pomiędzy wektorami, wartość odległości osiąga nieskończoność. Na uwagę zasługuje również fakt, iż uzyskany wynik nie jest odległością w sensie matematycznym, ponieważ funkcja ta nie spełnia zasady trójkąta.
Podobny kształt funkcji odległości uzyskiwany jest dla funkcji trapezoidalnej, z tą różnicą, że obszar pełnej przynależności równoważny jest zerowej wartości odległości.
Innym przykładem często stosowanego operatora T-normy jest operator minimum. Granica decyzji systemu rozmytego, bazującego na takim operatorze równoważna jest odległości Czebyszewa, przy czym składniki odległości można przetransformować wykorzystując dowolną monotoniczną postać transformacji $T(\cdot)$. Ten typ odległości stosowany jest również w systemach reguł-P w celu uzyskania zbioru reguł ostrych.
Dla T-normy Łukasiewicza (9.12) odpowiednią transformacją do postaci znormalizowanej odległości jest transformacja (9.2). Proces transformacji został pokazany w (9.13)

$$S(\mathbf{x},\mathbf{p}) = \max\left(\mu_1(x_1|p_1) + \mu_2(x_2|p_2) - 1, 0\right) \tag{9.12}$$

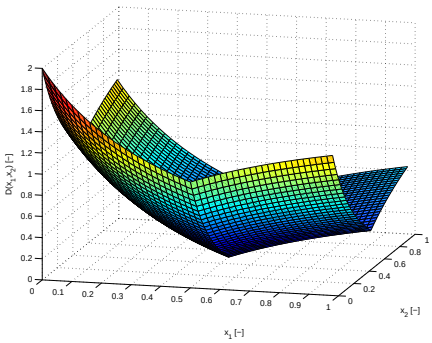
$$\begin{aligned} &D(\mathbf{x},\mathbf{p}) = 1 - \max\left(\mu_1(x_1|p_1) + \mu_2(x_2|p_2) - 1, 0\right) = \\ &\min\left((1-\mu_1(x_1|p_1)) + (1-\mu_2(x_2|p_2)), 1\right) \end{aligned} \tag{9.13}$$

9.4.2 Od funkcji odległości do funkcji przynależności

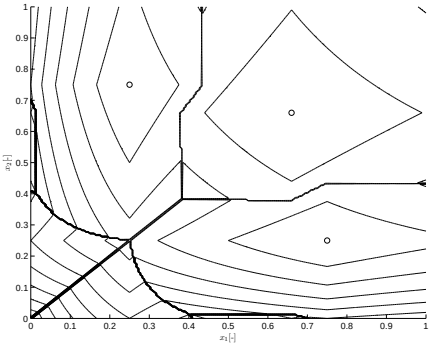
Równie ciekawym procesem jest transformacja funkcji odległości do postaci funkcji przynależności. Ponieważ większość funkcji odległości ma postać addytywną, dlatego też odpowiednią transformacją $T(\cdot)$ jest (9.4). Transformacja pomiędzy odległości Euklidesa bądź też innymi odległościami Minkowskiego a odpowiednimi funkcjami przynależności została opisana w poprzednim podrozdziale, a zarazem jest ona dosyć oczywista. Dosyć ciekawą funkcją odległości jest odległość Canberra (5.6) - ponieważ również ona jest addytywną funkcją odległości transformacji dokonuje się poprzez (9.4), co zostało pokazane jako (9.14).

$$\begin{aligned} &S(\mathbf{x},\mathbf{p}) = \exp\left(D(\mathbf{x},\mathbf{p})^q\right) = \exp\left(\sum_{i=1}^{n}\left|\frac{x_i-p_i}{x_i+y_i}\right|\right) = \prod_{i=1}^{n}\exp\left(\left|\frac{x_i-p_i}{x_i+p_i}\right|\right) \\ &\mu\left(x_i;p_i\right) = \exp\left(\left|\frac{x_i-p_i}{x_i+p_i}\right|\right) \end{aligned} \tag{9.14}$$

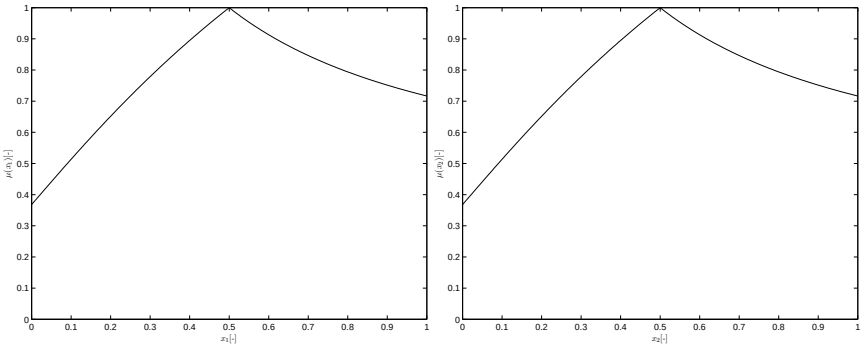
Kształt uzyskanej funkcji przynależności przedstawia rys.(9.2) Innym przykładem addytywnej funkcji odległości jest odległości VDM, która jest jedną z metod powszechnie stosowanych przy analizie danych symbolicznych. Jej transformacja do postaci funkcji przynależności częściowo rozwiązuje problem analizy danych symbolicznych, co stanowiło poważny problem w przypadku reguł-F. Proces transformacji odległości VDM



(a) Funkcja odległości



(b) Przykładowa granica decyzji dla trzech prototypów lub trzech reguł rozmytych



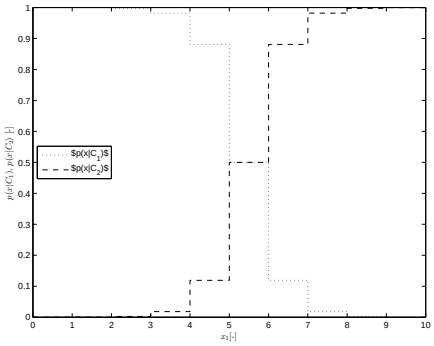
(c) Funkcje przynależności

Rysunek 9.2: Odległość Canberra oraz odpowiadająca jej funkcja przynależności

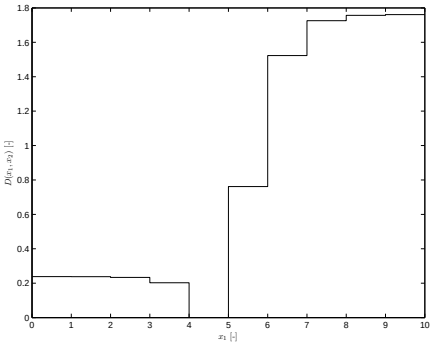
do postaci iloczynu funkcji przynależności przedstawia przekształcenie (9.15).

$$
\begin{aligned}
S(\mathbf{x},\mathbf{p}) &= \exp\left(-D(\mathbf{x},\mathbf{p})^q\right) = \exp\left(-\sum_{i=1}^{n}\sum_{j=1}^{c}|p(C_j|x_i)-p(C_j|p_i)|^q\right) \\
S(\mathbf{x},\mathbf{p}) &= \prod_{i=1}^{n}\exp\left(-\sum_{j=1}^{c}|p(C_j|x_i)-p(C_j|p_i)|^q\right) \\
\mu(x_i|p_i) &= \exp\left(-\sum_{j=1}^{c}|p(C_j|x_i)-p(C_j|p_i)|^q\right)
\end{aligned}
\tag{9.15}
$$

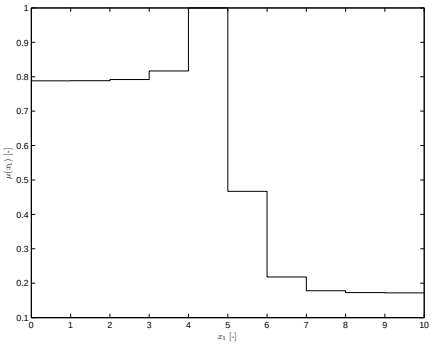
Uzyskana postać funkcji przynależności nie posiada prostej interpretacji lingwistycznej (rys.(9.3)), jednakże można ją traktować jako naturalny kształt przynależności do danej wartości symbolicznej.
Cechą charakterystyczną i naturalną dla systemów rozmytych jest różnorodność funkcji przynależności stosowanych dla różnych atrybutów. Ta różnorodność w przypadku reguł-P uzyskiwana jest poprzez wykorzystanie odległości heterogenicznych. Rezultatem transformacji odległości heterogenicznych do postaci iloczynu funkcji przynależności jest więc równoważna przypisaniu różnych typów funkcji przynależności do odpowiednich atrybutów.



(a) Rozkład $p(x|C_1)$ oraz $p(x|C_2)$



(b) Funkcja odległości



(c) Funkcja przynależności wyznaczona na podstawie odległości VDM

Rysunek 9.3: Odległość VDM oraz odpowiadająca jej funkcja przynależności dla prototypu położonego w $\mathbf{p} = [4]$

9.5 Wnioski z równoważności systemów

Przedstawione możliwości wzajemnej transformacji przesłanek reguł prototypowych oraz rozmytych pozwalają na otwarcie nowych możliwości oraz funkcjonalności dla każdej z metod. Głównym zyskiem uzyskanym dla systemu reguł-P jest szereg nowych miar odległości nie stosowanych dotychczas w celu zrozumienia danych. Podobnie dla reguł-F przedstawiono nowe możliwości kształtów funkcji przynależności, w szczególności rozpatrując problem analizy cech symbolicznych. Odległość VDM stanowi tu ciekawe rozwiązanie, pozwalające na uzyskanie naturalnego kształtu funkcji przynależności dla cech symbolicznych.
Kolejną istotną cechą wynikającą z równoważności obydwu postaci reguł jest możliwość dostosowania wydobytej wiedzy do wymagań eksperta analizującego dany problem. W zależności od jego preferencji istnieje możliwość wzajemnej transformacji obydwu typów systemów do postaci równoważnych. Jest to tym bardziej istotne, iż dotychczas obydwa typy systemów najczęściej rozwijane były niezależnie, co spowodowało różnorodność rozwiązań stosowanych w obydwu typach reguł. Większość z przedstawionych w rozdziale (6) metod selekcji prototypów nie była dotychczas stosowana w procesie

ekstrakcji reguł oraz rozumienia danych. Równoważność obydwu typów systemów pozwala więc na reprezentację wiedzy pochodzącą z metod selekcji prototypów do postaci zbioru reguł rozmytych.
Bardzo istotną pochodną omówionych tutaj metod transformacji jest integracja podobieństwa z logiką. Ma to szczególne znaczenie dla badań kognitywnych, pozwalając na „logiczną" analizę i interpretację uzyskanych wyników badań.

Rozdział 10

Wydobywanie reguł prototypowych z danych

10.1 Opis modeli

Analiza przedstawionych w rozprawie różnych metod selekcji i optymalizacji prototypów oraz selekcji cech pozwoliła na określenie modeli, które posłużyły budowie systemu ekstrakcji reguł prototypowych o nazwie SBPS (ang. similarity-based prototype system). System ten składa się z kilku omówionych w pracy metod, jednakże do praktycznej analizy danych wykorzystano dwa zaproponowane w pracy autorskie rozwiązania.
Wyniki zamieszczone w rozdziale 6 jednoznacznie wskazują na wysoką jakość zaproponowanej w pracy adaptacji algorytmu warunkowego grupowania danych do selekcji wektorów prototypowych oraz algorytmu wyścigu optymalizującego liczbę tych wektorów. Dlatego też integracja tego algorytmu z metodami selekcji cech wydaje się przynieść największe korzyści. Ze względu na niską złożoność obliczeniową obydwu algorytmów: CFCM oraz LVQ możliwe jest zastosowanie wszystkich metod selekcji cech, w tym bazujących na przeszukiwaniu.
Zaprezentowane w rozdziale 8.1.3 wyniki nie dają jednoznacznej odpowiedzi, która z metod selekcji jest najlepsza, dlatego też sugerowanym rozwiązaniem jest jednoczesne uwzględnienie różnych metod selekcji cech w celu wybrania takiego, który zarówno maksymalizacji dokładności, jak również minimalizacji złożoności modelu. Jako selektory cech dla metody CFCM+LVQ używane były algorytmy *selekcji w przód*, *selekcji w tył* oraz metoda drzew. W zaprezentowanych poniżej wynikach pominięto metody selekcji bazujące na rankingach, z uwagi na niedużą liczbę cech analizowanych danych (maksymalnie 60). Należy zaznaczyć, iż dla dużych zbiorów o znacznej liczbie cech (przekraczających 100) sugerowanym rozwiązaniem jest selekcja cech w oparciu o algorytm drzew decyzji lub metody rankingowe.
W przypadku drugiego algorytmu ekstrakcji reguł prototypowych progowych autor posłużył się algorytmem OPTDL. Porównanie metody OPDL z heterogenicznymi drzewami bazującymi jedynie na macierzy odległości nie wskazuje jednoznacznie na zwycięzcę, a obydwa algorytmy cechuje porównywalna dokładność. Ponadto na korzyść algorytmu OPTDL przemawia płaska struktura uzyskanych reguł. Ponieważ algorytm OPDL ma złożoność rzędu $O(m^2)$, dlatego też konieczny jest wybór metody selekcji cech o małej złożoności. W zaprezentowanych poniżej wynikach zostały wykorzystane algorytmy rankingowe opakowujące, bazujące na analizie współczynnika korelacji

linowej oraz metoda drzew.
Ponieważ istotą systemów regułowych jest łatwość zrozumienia uzyskanych wyników, dlatego też w praktyce zastosowano miary odległości z rodziny HVDM. Realizowane rzutowanie do przestrzeni prawdopodobieństwa odbywa się jedynie dla cech symbolicznych, dyskretnych oraz binarnych (uciąglanie cech w oparciu o miarę VDM), natomiast cechy ciągłe pozostają nie zmienione poddane jedynie normalizacji lub standaryzacji. Pominięcie w praktycznych obliczeniach zaproponowanych miar heterogenicznych dla atrybutów ciągłych związane jest z łatwością interpretacji uzyskanych wyników obliczeń.
Algorytm OPDL nie wymaga optymalizacji parametrów modelu, gdyż proces ten realizowany jest na bazie wewnętrznej walidacji krzyżowej, gdzie dobierana jest optymalna liczba prototypów. Natomiast maksymalne wykorzystanie algorytmu CFCM+LVQ zmusza do przeszukania przestrzeni modelu w celu znalezieniu odpowiednich nastaw. Na przestrzeń modelu składają się tutaj współczynniki nagrody i kary algorytmu LVQ czy też współczynniki filtru Gaussowskiego. Proces ten wraz z problemem doboru metody selekcji cech realizowany jest na bazie omówionych w podrozdziale 8.2 metod przeszukiwania, gdzie testowane są kolejne modele z różnymi wartościami nastaw.

10.2 Wyniki uzyskane przez systemy reguł prototypowych

Do analizy danych posłużono się dziewięcioma zbiorami pochodzącymi z repozytorium UCI [117] oraz zbiorem Lancet uzyskanym od Walkera, Crossa oraz Harrisona [171]. Krótki opis użytych zbiorów danych znajduje się w rozdziale 12. Wybór danych, w większości pochodzących z repozytorium UCI, podyktowany był powszechną dostępnością źródła danych, co pozwala na porównanie rezultatów uzyskanych przez system SBPS z wynikami innych badaczy. Kryterium doboru zbiorów danych była ich różnorodność zarówno pod względem liczebności przypadków (zbiory liczyły od 106 do 768 wektorów), jak również pod względem różnorodności typów oraz liczby cech (od 4 do 60 atrybutów).
Procedura testowa polegała na 10-cio krotnym teście krzyżowym, przy czym zamieszczone wyniki porównawcze są rezultatami walidacji całości modelu. Na każdy z modeli składały się cztery etapy przetwarzania danych. W etapie pierwszym dane były normalizowane lub standaryzowane, następnie dokonywana była procedura uciąglania cech symbolicznych i dyskretnych z wykorzystaniem metryki VDM, po której zbiór danych poddawany był procesowi selekcji cech w oparciu o algorytm opakowywania lub algorytm drzew oraz ostatecznie następowało uczenie klasyfikatora. W teście użyte były modele opisane w poprzednim podrozdziale. Zamieszczone w tabelach wyniki stanowią średnią dokładność (*Acc*) oraz odchylenie standardowe. Do każdej tabeli dodano również informację o złożoności modelu (A/R) określoną jako liczba wybranych cech (A) oraz liczba reguł (R). Dodatkowo dla każdego zbioru danych zamieszczono uzyskany zbiór prototypów wraz z indeksami wybranych przez model cech. Rezultaty te są wynikiem uczenia modelu na całym zbiorze danych.
Różnice wyników pomiędzy rezultatami zamieszczonymi w rozdziale 8.1.3, a przedstawionymi poniżej wynikają z różnic procedury testowej. W tabeli 8.1.3

Prototypy:									
Cecha:	f_2^{VDM}	f_3^{VDM}	f_6^{VDM}	f_7^{VDM}	f_5^{VDM}	f_8	f_4^{VDM}		Θ
$\mathbf{P}_1 = [$	0.8222	0.8793	0.9627	0.7764	0.9446	0.8333	0.6379	$]^T$	1.0474

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ mniej niż Θ_1 **to** C($\mathbf{X}$)=C_1
jak nie, C($\mathbf{X}$)=C_2

Tablica 10.1: Zbiór reguł oraz położenia prototypów opisujących zbiór *rak piersi*. Algorytm OPTDL

Prototypy:				
Cecha:	f_3^{VDM}	f_4^{VDM}	f_7^{VDM}	
$\mathbf{P}_1 = [$	0.5245	0.4059	0.1024	$]^T$
$\mathbf{P}_2 = [$	0.0328	0.0788	0.1665	$]^T$
$\mathbf{P}_3 = [$	0.9525	0.9453	0.9259	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C($\mathbf{X}$)=C_1

Tablica 10.2: Zbiór reguł oraz położenia prototypów opisujących zbiór *rak piersi*. Algorytm CFCM+LVQ

zamieszczono wyniki testu krzyżowego, któremu poddany był jedynie klasyfikator, natomiast proces selekcji cech realizowany był niezależnie od procesu walidacji. Poniżej zamieszczone wyniki stanowią rezultaty walidacji całości systemu, gdzie poszczególne etapy przetwarzania zostały zagnieżdżone w procesie testu krzyżowego.
Zamieszczone wyniki innych metod ekstrakcji reguł uzyskano wykorzystując do obliczeń pakiety Weka [186] oraz NefClass [127], jak również na podstawie wyników innych autorów.

10.2.1 Rak piersi

Dla zbioru *raka piersi* obydwie metody prototypowe dają porównywalne rezultaty średniej dokładności, co wraz z porównaniem z innymi algorytmami ekstrakcji reguł przedstawia tabela (10.3). Godnym uwagi jest fakt, iż średnie dokładności obydwu zaprezentowanych metod są tutaj największe i przewyższają wyniki metod konkurencyjnych.
Rozwiązanie znalezione przez algorytm OPTDL uzyskano stosując algorytm selekcji cech bazujący na metodzie rankingowej selekcji w przód z wykorzystaniem współczynnika korelacji liniowej. Na zbiorze treningowym algorytm OPTDL uzyskał dokładność 97.65% przypadków poprawnie klasyfikowanych. Uzyskane rozwiązanie opisane w postaci pojedynczego prototypu dla klasy C_1 (tab.(10.1)) wskazuje, iż granica decyzji pomiędzy nowotworem złośliwym a łagodnym ma kształt sferyczny.
Podobnie proste rozwiązanie znalazł algorytm CFCM+LVQ, w którym wykorzystano algorytm selekcji cech w oparciu o drzewa decyzji. Znalezione rozwiązanie opisane w postaci trzech cech oraz trzech prototypów możne wskazywać, iż sferyczny kształt granicy można aproksymować w postaci dwóch półprostych.

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	96.79 ± 2.02	3/3	SBPS
OTPDL	96.64 ± 2.06	7/1	SBPS
C4.5	95.44 ± 2.64	6/11	Weka
Decision Table	95.36 ± 2.34	3/20	Weka
OneR	92.15 ± 2.96	1/1	Weka
NefClass	92.19 ± 2.48	2/2	NefClass

Tablica 10.3: Zbiór: *rak piersi*. Wyniki testu 10xCV

Prototypy:			
Cecha:	f_4	f_7	
$\mathbf{P}_1 = [$	0.2533	0.4464	$]^T$
$\mathbf{P}_2 = [$	0.0760	0.0780	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C(**X**)=C_1
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C(**X**)=C_2

Tablica 10.4: Zbiór reguł oraz położenia prototypów opisujących zbiór *wyrostek robaczkowy*. Algorytm CFCM+LVQ

10.2.2 Wyrostek robaczkowy

Rozwiązania znalezione przez zaproponowane w pracy algorytmy cechowały podobne rezultaty w porównaniu z rozwiązaniami konkurencyjnymi (tab.(10.6)). Należy jednak zwrócić uwagę na dużą wartość odchylenia standardowego uzyskanych wyników. Wskazuje ona, iż dla części przypadków walidacji algorytm uzyskiwał bardzo dużą dokładność, a dla części uzyskane wyniki były znacząco gorsze. Na jednak zwrócić uwagę na małą liczebność zbioru oraz silne niezbalansowanie liczby wektorów z poszczególnych klas. Właściwości algorytmu wyścigu stosowanego do optymalizacji liczby prototypów w przypadku metody CFCM+LVQ cechuje dążenie do porównywalnej dokładności klasyfikacji poszczególnych klas, dlatego też całkowita dokładność klasyfikacji tego algorytmu wypada stosunkowo nisko. Analizując jednak opublikowane przez autora wyniki porównawcze opisane w [17], gdzie kryterium optymalizacji była dokładność zbalansowana (BAcc) omawiany algorytm znacznie przewyższył jakość uzyskanych wyników metod konkurencyjnych. W przypadku algorytmu CFCM+LVQ zastosowanym kryterium selekcji cech była metoda drzew decyzji. Uzyskany zbiór dwóch prototypów (tab.(10.4)) wskazuje, iż optymalną granicą decyzji jest funkcja liniowa wyznaczona przez dwa prototypy.

Uzyskany rezultat dla algorytmu OPTDL został osiągnięty poprzez wykorzystanie algorytmu selekcji cech bazującego na metodzie rankingowej - selekcji w przód, z kryterium jakim był współczynnik korelacji liniowej. Wyznaczone położenie prototypu opisuje tabela (10.5), który pozwalał na klasyfikację zbioru treningowego z dokładnością 91.51%.

10.2.3 Cukrzyca

Dla tego zbioru algorytm OPTDL wyznaczył dwa prototypy opisane w tabeli (10.7). Pozwalają one na klasyfikację z dokładnością 76.56%. W przypadku obydwu algorytmów za optymalny algorytm selekcji cech została wybrana metoda drzew decyzji. Analiza zbioru algorytmem CFCM+LVQ, dały najlepsze rezultaty w porównaniu z innymi metodami (tab.(10.9)). Wynikiem analizy tego zbioru w oparciu o reguły typu k-NNsą dwa prototypy przedstawione w tabeli (10.8), przy czym położenie prototypów jest

Prototypy:								
Cecha:	f_2	f_3	f_6	f_7	f_1	f_5		Θ
$\mathbf{P}_1 = [$	0.4140	0.6456	0.3440	0.6045	0.6346	2.4456	$]^T$	19.4864

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ mniej niż Θ_1 **to** C(**X**)=C_1
jak nie, C(**X**)=C_2

Tablica 10.5: Zbiór reguł oraz położenia prototypów opisujących zbiór *wyrostek robaczkowy*. Algorytm OPTDL

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	83.77 ± 18.37	2/2	SBPS
OTPDL	83.88 ± 11.23	6/1	SBPS
C4.5	84.55 ± 10.47	2/2	Weka
Decision Table	85.76 ± 9.14	3/4	Weka
OneR	83.53 ± 11.11	1/3	Weka
NefClass	87.83 ± 6.38	1/2	NefClass

Tablica 10.6: Zbiór: *zapalenie wyrostka robaczkowego*. Wyniki testu 10xCV

Prototypy:				
Cecha:	f_2	f_6		Θ
$\mathbf{P}_1 = [$	2.0676	1.4722	$]^T$	6.6483
$\mathbf{P}_2 = [$	-0.6535	1.0791	$]^T$	0.7569

Reguły:
jeżeli X jest podobny do $\mathbf{P}_2$ mniej niż Θ_2 **to** C(**X**)=C_1
jak nie, jeżeli X jest podobny do $\mathbf{P}_1$ bardziej niż Θ_1 **to** C(**X**)=C_2
jak nie, C(**X**)=C_1

Tablica 10.7: Zbiór reguł oraz położenia prototypów opisujących zbiór *cukrzyca*. Algorytm OPTDL

Prototypy:			
Cecha:	f_2	f_6	
$\mathbf{P}_1 = [$	0.9309	0.6250	$]^T$
$\mathbf{P}_2 = [$	0.4621	0.3851	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C(**X**)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C(**X**)=C_1

Tablica 10.8: Zbiór reguł oraz położenia prototypów opisujących zbiór *cukrzyca*. Algorytm CFCM+LVQ

określone po uprzedniej normalizacji danych wejściowych. Uzyskany rezultat wskazuje, iż kształt granicy decyzji dla tego zbioru przyjmuje postać hiperpłaszczyzny separującej obydwie klasy.

10.2.4 Sonar

Wyniki porównawcze dla tego zbioru przedstawia tabela (10.12). Wskazują one na gorsze właściwości zaproponowanych w pracy algorytmów porównując jedynie uzyskane dokładności klasyfikacji, jednak na uwagę zasługuje prostota znalezionych rozwiązań.

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	76.96 ± 3.7	2/2	SBPS
OTPDL	75.13 ± 5.38	2/2	SBPS
C4.5	74.49 ± 5.27	7/20	Weka
Decision Table	74.08 ± 4.40	5/32	Weka
OneR	72.00 ± 4.72	1/8	Weka
NefClass	74.86 ± 3.00	1/2	NefClass

Tablica 10.9: Zbiór: *cukrzyca*. Wyniki testu 10xCV

Prototypy:		
Cecha:	f_{11}	Θ
$\mathbf{P}_1 = [$	$0.4918 \quad]^T$	0.0863

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ mniej niż Θ_1 **to** C(**X**)=C_2
jak nie, C(**X**)=C_1

Tablica 10.10: Zbiór reguł oraz położenia prototypów opisujących zbiór *sonar*. Algorytm OPTDL

Prototypy:	
Cecha:	f_{11}
$\mathbf{P}_1 = [$	$0.4449 \quad]^T$
$\mathbf{P}_2 = [$	$0.0631 \quad]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C(**X**)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C(**X**)=C_1

Tablica 10.11: Zbiór reguł oraz położenia prototypów opisujących zbiór *sonar*. Algorytm CFCM+LVQ

Metoda optymalizacji modelu, za najlepszy algorytm selekcji cech wybrała algorytm drzewa decyzji, który wybrał zaledwie pojedynczą cechę do klasyfikacji. Do rozwiązania problemu dyskryminacji sygnałów radarowych wystarczyła więc pojedyncza reguła dla algorytmu OPTDL (tab.(10.10)) oraz dwa prototypy algorytmu CFCM+LVQ (tab.(10.11))

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	66.53 ± 17.81	1/2	SBPS
OTPDL	65.53 ± 14.27	1/1	SBPS
C4.5	71.17 ± 7.10	13/17	Weka
Decision Table	74.50 ± 8.17	7/37	Weka
OneR	62.50 ± 11.07	1/4	Weka
NefClass	53.88 ± 13.97	1/1	NefClass

Tablica 10.12: Zbiór: *sonar*. Wyniki testu 10xCV

Prototypy:							
Cecha:	f_5	f_3	f_6	f_2	f_3		Θ
$\mathbf{P}_1 = [$	0.2000	0.0728	0.3636	0.1712	0.2000	$]^T$	0.1120
$\mathbf{P}_2 = [$	0.6348	0.1987	0.5714	0.5616	0.2000	$]^T$	0.3515
$\mathbf{P}_3 = [$	0.2522	0.4768	0.3506	0.0651	0.4000	$]^T$	0.1438
$\mathbf{P}_4 = [$	0.3043	0.1788	0.1558	0.0411	0.1000	$]^T$	0.0146
Reguły:							
jeżeli X jest podobny do $\mathbf{P}_4$ mniej niż Θ_4 **to** C(**X**)=C_1							
jak nie, jeżeli X jest podobny do $\mathbf{P}_3$ bardziej niż Θ_3 **to** C(**X**)=C_1							
jak nie, jeżeli X jest podobny do $\mathbf{P}_2$ bardziej niż Θ_2 **to** C(**X**)=C_2							
jak nie, jeżeli X jest podobny do $\mathbf{P}_1$ bardziej niż Θ_1 **to** C(**X**)=C_2							
jak nie, C(**X**)=C_1							

Tablica 10.13: Zbiór reguł oraz położenia prototypów opisujących zbiór *choroby wątroby*. Algorytm OPTDL

Prototypy:				
Cecha:	f_3	f_4	f_5	
$\mathbf{P}_1 = [$	0.5501	-0.9789	-0.2369	$]^T$
$\mathbf{P}_2 = [$	-0.5021	0.2021	0.2392	$]^T$
$\mathbf{P}_3 = [$	0.4819	-0.7283	-0.7005	$]^T$
Reguły:				
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C(**X**)=C_1				
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C(**X**)=C_2				
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C(**X**)=C_1				

Tablica 10.14: Zbiór reguł oraz położenia prototypów opisujących zbiór *choroby wątroby*. Algorytm CFCM+LVQ

10.2.5 Choroby wątroby

Wyniki porównawcze przedstawione w tab.(10.15) wskazują, iż jedynie algorytm drzewa decyzji C4.5 uzyskał lepsze rezultaty niż zaproponowane przez autora rozwiązania dystansując inne metody. Zaskakująco duża jest jednak liczba reguł znalezionych przez algorytm C4.5. Uzyskane w oparciu o metodę CFCM+LVQ rozwiązanie opisane jest przez trzy prototypy oraz trzy cechy (tab.(10.14)), co świadczy o prostocie znalezionego rozwiązania. Wybór cech został tutaj dokonany w oparciu o metodę selekcji w przód. Rozwiązanie znalezione przez algorytm OPTDL wymaga większej liczby prototypów oraz liczby cech (tab.(10.13)). Algorytm drzewa decyzji wybrał tutaj pięć cech pozwalających na klasyfikację zbioru treningowego z dokładnością 75.07%.

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	66.40 ± 7.87	3/3	SBPS
OTPDL	65.82 ± 6.21	5/4	SBPS
C4.5	68.71 ± 8.74	6/25	Weka
Decision Table	57.72 ± 5.64	2/3	Weka
OneR	54.82 ± 7.26	1/14	Weka
NefClass	57.08 ± 0.722	1/1	NefClass

Tablica 10.15: Zbiór: *chorób wątroby*. Wyniki testu 10xCV

Prototypy:				
Cecha:	f_3	f_4		Θ
$\mathbf{P}_1 = [$	14	2	$]^T$	0.1215
$\mathbf{P}_2 = [$	14	2	$]^T$	0.6594

Reguły:
jeżeli X jest podobny do $\mathbf{P}_2$ mniej niż Θ_2 **to** C($\mathbf{X}$)=C_3
jak nie, jeżeli X jest podobny do $\mathbf{P}_1$ bardziej niż Θ_1 **to** C($\mathbf{X}$)=C_1
jak nie, C($\mathbf{X}$)=C_2

Tablica 10.16: Zbiór reguł oraz położenia prototypów opisujących zbiór *irysy*. Algorytm OPTDL

Prototypy:			
Cecha:	f_3	f_4	
$\mathbf{P}_1 = [$	0.5441	0.5001	$]^T$
$\mathbf{P}_2 = [$	0.8176	0.8980	$]^T$
$\mathbf{P}_3 = [$	0.7384	0.7256	$]^T$
$\mathbf{P}_4 = [$	0.0784	0.0610	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_3
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C($\mathbf{X}$)=C_3
jeżeli X jest podobny do $\mathbf{P}_4$ **to** C($\mathbf{X}$)=C_1

Tablica 10.17: Zbiór reguł oraz położenia prototypów opisujących zbiór *irysy*. Algorytm CFCM+LVQ

10.2.6 Irysy

Dlatego tego zbioru niemal wszystkie metody charakteryzują się porównywalną i bardzo dużą dokładnością (tab.(10.18)). Obydwa zaproponowane w pracy algorytmy reguł prototypowych wykorzystywały metodę selekcji cech bazującą na drzewach decyzji. Znalezione położenia prototypów po uprzedniej normalizacji danych przedstawiają tabele (10.17) oraz (10.16). Analiza wyników uzyskanych przez algorytm OPTDL wskazuje iż dwukrotnie wybrany został ten sam prototyp jednak z różną wartością progu. Należy tutaj zwrócić uwagę, iż algorytm OPTDL znacznie pobił inne konkurencyjne metody, natomiast algorytm CFCM+LVQ uplasował się na drugim miejscu.

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	96.67 ± 4.71	2/4	SBPS
OTPDL	98.00 ± 3.22	2/2	SBPS
C4.5	96.00 ± 5.62	2/4	Weka
Decision Table	92.67 ± 5.84	3/6	Weka
OneR	94.00 ± 4.92	1/3	Weka
NefClass	96.00 ± 4.59	1/2	NefClass

Tablica 10.18: Zbiór: *irysy*. Wyniki testu 10xCV

10.2.7 Winorośl

Wyniki porównawcze dla różnych metod przedstawia tabela (10.21). Najlepsze rozwiązanie dla porównania różnych metod zostało znalezione z wykorzystaniem algorytmu CFCM+LVQ (tab(10.20)), który prawie o 1.7% pobił drugi z algorytmów drzewo C4.5 przy mniejszej wariancji wyników i porównywalnej złożoności rozwiązania.

Prototypy:						
Cecha:	f7	f11	f12	f13		Θ
$\mathbf{P}_1 = [$	-0.0443	0.0976	0.1502	0.3937	$]^T$	0.1379
$\mathbf{P}_2 = [$	0.3565	0.5772	0.4432	0.0813	$]^T$	0.2187
$\mathbf{P}_3 = [$	0.0443	0.0976	0.1502	0.3937	$]^T$	0.1379
$\mathbf{P}_4 = [$	0.4937	0.3496	0.6337	0.5399	$]^T$	0.0715

Reguły:
jeżeli X jest podobny do $\mathbf{P}_4$ mniej niż Θ_4 **to** C($\mathbf{X}$)=C_3
jak nie, jeżeli X jest podobny do $\mathbf{P}_3$ bardziej niż Θ_3 **to** C($\mathbf{X}$)=C_2
jak nie, jeżeli X jest podobny do $\mathbf{P}_2$ bardziej niż Θ_2 **to** C($\mathbf{X}$)=C_3
jak nie, jeżeli X jest podobny do $\mathbf{P}_1$ bardziej niż Θ_1 **to** C($\mathbf{X}$)=C_1
jak nie, C($\mathbf{X}$)=C_1

Tablica 10.19: Zbiór reguł oraz położenia prototypów opisujących zbiór *winorośl*. Algorytm OPTDL

Cecha:	f_1	f_3	f_4	f_7	f_{11}	f_{13}	
$\mathbf{P}_1 = [$	-0.9450	-0.4403	0.2509	0.0528	0.4550	-0.7425	$]^T$
$\mathbf{P}_2 = [$	0.9229	0.3300	-0.7366	0.9611	0.4708	1.1903	$]^T$
$\mathbf{P}_3 = [$	0.2024	0.2772	0.5665	-1.2812	-1.2332	-0.3561	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_1
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C($\mathbf{X}$)=C_3

Tablica 10.20: Zbiór reguł oraz położenia prototypów opisujących zbiór *winorośl*. Algorytm CFCM+LVQ

Ponieważ zbiór ten składa się z trzech klas dlatego też minimalne rozwiązanie dla systemu reguł typu k-NNwymagało 3 prototypów, co oznacza że poszczególne klasy są liniowo separowalne. Kiepskie wyniki uzyskane przez algorytm OPTDL można tłumaczyć liniowym kształtem granicy decyzji pomiędzy poszczególnymi klasami. W obydwu przypadkach wykorzystano metody selekcji cech bazujące na algorytmie drzew decyzji.

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	95.54 ± 4.91	6/3	SBPS
OTPDL	88.81 ± 9.38	4/5	SBPS
C4.5	93.86 ± 5.52	3/4	Weka
Decision Table	91.05 ± 6.49	6/41	Weka
OneR	76.41 ± 8.98	1/6	Weka
NefClass	92.22 ± 8.045	1/3	NefClass

Tablica 10.21: Zbiór: *winorośl*. Wyniki testu 10xCV

10.2.8 Jonosfera

Bardzo dobrymi wynikami analizy charakteryzuje się algorytm OPTDL, pozwalając na klasyfikację zbioru treningowego z dokładnością 89.46% przy zaledwie jednej regule

Prototypy:				
Cecha:	f_3	f_{25}		Θ
$\mathbf{P}_1 = [$	0.9707	0.5665]	$]^T$	0.1883

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ mniej niż Θ_1 **to** C($\mathbf{X}$)=C_1
jak nie, C($\mathbf{X}$)=C_2

Tablica 10.22: Zbiór reguł oraz położenia prototypów opisujących zbiór *jonosfera*. Algorytm OPTDL

Prototypy:						
Cecha:	f_2	f_4	f_5	f_{21}	f_{26}	
$\mathbf{P}_1 = [$	-0.0730	-2.1542	-0.0885	-0.4869	0.4903	$]^T$
$\mathbf{P}_2 = [$	-1.8982	-0.8291	-0.0075	-0.1092	0.7909	$]^T$
$\mathbf{P}_3 = [$	0.3090	0.6721	2.0769	1.5144	0.9048	$]^T$
$\mathbf{P}_4 = [$	-1.5420	-1.6746	-0.3159	-0.2286	-1.515	$]^T$
$\mathbf{P}_5 = [$	0.4521	0.4076	0.1288	-0.1147	-0.0750	$]^T$
$\mathbf{P}_6 = [$	0.3576	0.3007	-2.2575	1.2493	0.8369	$]^T$

Reguły:
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_4$ **to** C($\mathbf{X}$)=C_2
jeżeli X jest podobny do $\mathbf{P}_5$ **to** C($\mathbf{X}$)=C_1
jeżeli X jest podobny do $\mathbf{P}_6$ **to** C($\mathbf{X}$)=C_2

Tablica 10.23: Zbiór reguł oraz położenia prototypów opisujących zbiór *jonosfera*. Algorytm CFCM+LVQ

i dwóch atrybutach (tab.(10.22)). Średnia dokładność testu krzyżowego jest jednak gorsza o ponad 2% w porównaniu z algorytmem C4.5 czy też algorytmem DecisionTable (tab.(10.24)). Rozważając jednak prostotę znalezionego rozwiązania, algorytm ten wysuwa się na prowadzenie, gdyż rezultat osiągnięty przez algorytm C4.5 wymagał aż 14 atrybutów oraz 18 reguł. Jako metoda selekcji cech posłużył tutaj algorytm drzew decyzji.
Średnia dokładność wyników znalezionych dla reguł typu k-NNjest porównnywala z dokładnością algorytmu OPTDL przy jednak mniejszym odchyleniu standardowym uzyskanych wyników. Znalezione przez metodę CFCM+LVQ rozwiązanie wymagało 6 prototypów opisanych na 5 atrybutach znalezionych przez algorytm *selekcji w przód* (tab.(10.23)).

10.2.9 Lancet

Dla zbioru *lancet* algorytm OPTDL znalazł opis składający się z dwóch reguł opisanych prototypami w tab.(10.25) oraz 3 cech. System ten pozwala na uzyskanie dokładności zbioru treningowego 92.77%

Analiza uzyskanych rezultatów wskazuje, iż podobnie jak w przypadku zbioru *raka piersi* kształt granicy decyzji przyjmuje kształt (hipersfery) o czym świadczy duża wartość progu dla prototypu $\mathbf{P}_1$. Powoduje to, że opis tego zbioru w oparciu o algorytm

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	87.82 ± 4.47	5/6	SBPS
OTPDL	87.57 ± 6.99	2/1	SBPS
C4.5	89.74 ± 4.38	14/18	Weka
Decision Table	88.97 ± 5.19	3/22	Weka
OneR	82.39 ± 5.44	1/7	Weka
NefClass	86.6 ± 4.13	3/4	NefClass

Tablica 10.24: Zbiór: *jonosfera*. Wyniki testu 10xCV

Prototypy:					
Cecha:	f_9^{VDM}	f_3^{VDM}	f_{11}^{VDM}		Θ
$\mathbf{P}_1 = [$	0.2293	0.0781	0.0238	$]^T$	0.5773
$\mathbf{P}_2 = [$	0.2293	0.8615	0.0238	$]^T$	0.0565
Reguły:					
jeżeli X jest podobny do $\mathbf{P}_2$ mniej niż Θ_2 **to** C($\mathbf{X}$)=C_2					
jak nie, jeżeli X jest podobny do $\mathbf{P}_1$ bardziej niż Θ_1 **to** C($\mathbf{X}$)=C_2					
jak nie, C($\mathbf{X}$)=C_1					

Tablica 10.25: Zbiór reguł oraz położenia prototypów opisujących zbiór *lancet*. Algorytm OPTDL

CFCM+LVQ jest bardziej złożony i wymaga czterech prototypów przedstawionych w tab.(10.26).

Prototypy:				
Cecha:	f_3	f_9	f_{11}	
$\mathbf{P}_1 = [$	0.2293	0.8615	0.4316	$]^T$
$\mathbf{P}_2 = [$	0.8796	0.0782	0.4051	$]^T$
$\mathbf{P}_3 = [$	0.9107	0.8615	0.364	$]^T$
$\mathbf{P}_4 = [$	0.2293	0.0781	0.3443	$]^T$
Reguły:				
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_1				
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_1				
jeżeli X jest podobny do $\mathbf{P}_3$ **to** C($\mathbf{X}$)=C_1				
jeżeli X jest podobny do $\mathbf{P}_4$ **to** C($\mathbf{X}$)=C_2				

Tablica 10.26: Zbiór reguł oraz położenia prototypów opisujących zbiór *lancet*. Algorytm CFCM+LVQ

10.2.10 Choroby serca

Analiza wyników tabeli (10.30) wskazuje, iż algorytm OPTDL zdystansował inne metody. Pojedynczy prototyp opisany w przestrzeni 3 cech (tab.(10.28)) pozwalał na średnią klasyfikację z dokładnością 84.16 ± 5.73 co o ponad 6% przewyższyło dokładność algorytmu C.45, o 2.25% system regułowy DecisionTable oraz 4% system rozmyty NefClass. Uzyskane wyniki wskazują, iż kształt granicy decyzji jest sferyczny. Na identycznym podzbiorze cech, wyznaczonym zarówno przez algorytm *selekcji w przód* bazujący na wartościach rankingowych dla algorytmu OPTDL, jak i na przeszukiwaniu

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	91.31 ± 3.46	3/4	SBPS
OTPDL	93.20 ± 2.73	3/2	SBPS
C4.5	95.16 ± 2.59	5/8	Weka
Decision Table	93.63 ± 2.78	6/20	Weka
OneR	88.89 ± 3.39	1/1	Weka
NefClass	90.19 ± 3.55	1/2	NefClass

Tablica 10.27: Zbiór: *lancet*. Wyniki testu 10xCV

zachłannym dla algorytmu CFCM+LVQ, druga z metod uzyskała nieco gorszy wynik średniej dokładności przy dwóch prototypach (tab.(10.29)).

Prototypy:					
Cecha:	f_{13}^{VDM}	f_7^{VDM}	f_12^{VDM}		Θ
$\mathbf{P}_1 = [$	0.77439	0.78313	0.74138	$]^T$	0.318
Reguły:					
jeżeli X jest podobny do $\mathbf{P}_1$ mniej niż Θ_1 **to** C($\mathbf{X}$)=C_1					
jak nie, C($\mathbf{X}$)=C_2					

Tablica 10.28: Zbiór reguł oraz położenia prototypów opisujących zbiór *choroby serca*. Algorytm PTDL

10.3 Podsumowanie wyników

Uzyskane w trakcie testów wyniki wskazują, iż metody reguł prototypowych stanowią poważną alternatywę do innych znanych metod wydobywania reguł z danych.

Prototypy:				
Cecha	f_3^{VDM}	f_{12}^{VDM}	f_{13}^{VDM}	
$\mathbf{P}_1 = [$	0.7232	0.7098	0.7603	$]^T$
$\mathbf{P}_2 = [$	0.3101	0.2974	0.3235	$]^T$
Reguły:				
jeżeli X jest podobny do $\mathbf{P}_1$ **to** C($\mathbf{X}$)=C_1				
jeżeli X jest podobny do $\mathbf{P}_2$ **to** C($\mathbf{X}$)=C_2				

Tablica 10.29: Zbiór reguł oraz położenia prototypów opisujących zbiór *choroby serca*. Algorytm CFCM+LVQ

Metoda	Dokładność [%]	A/R	Źródło
CFCM+LVQ	82.80 ± 6.83	3/2	SBPS
OTPDL	84.16 ± 5.73	3/1	SBPS
C4.5	77.75 ± 6.72	11/17	Weka
Decision Table	81.91 ± 7.02	4/8	Weka
OneR	72.23 ± 6.60	1/1	Weka
NefClass	79.82 ± 9.15	1/2	NefClass

Tablica 10.30: Zbiór: *choroby serca*. Wyniki testu 10xCV

Uzyskane wyniki często znacznie przewyższają rezultaty uzyskane metodami ekstrakcji klasycznych reguł ostrych oraz często również reguł rozmytych. Dla zbiorów *raka piersi* oraz *chorób serca* uzyskane w oparciu o algorytm OPTDL wyniki pozwalają na zapis wydobytej wiedzy w postaci zaledwie jednej reguły, dając jednocześnie bardzo dużą dokładność klasyfikacji przewyższającą rezultaty metod reguł ostrych oraz rozmytych zarówno pod względem dokładności, jak i skomplikowania struktury wydobytej wiedzy. Zastosowanie algorytmu CFCM+LVQ również pozwoliło na reprezentację wydobytej wiedzy w postaci bardzo prostych zbiorów reguł. Zaledwie dwa lub trzy prototypy często umożliwiają klasyfikację danych przewyższającą inne metody ekstrakcji reguł. Dowodzą tego rezultaty uzyskane m.in dla zbiorów *cukrzyca*, *rak piersi* oraz *choroby wątroby*.

Dla niektórych zbiorów danych jak np. *sonar* uzyskane wyniki odbiegają od rezultatów innych metod. Można to tłumaczyć prostotą uzyskanych wyników jak również niepełnym przeszukaniem przestrzeni modelu. Zarówno w przypadku reguł progowych, jak i k-NNużywana była funkcja odległości Euklidesa, podczas gdy jej modyfikacja (zmiana wykładnika potęgi) mogłaby przynieść oczekiwane rezultaty podniesienia dokładności klasyfikacji. Należy tutaj jednak zwrócić uwagę, iż istotą problemu jest zarówno dokładność klasyfikacji, jak i reprezentacja wydobytej wiedzy w najprostszy możliwy sposób.

Podsumowując należy zaznaczyć, iż obydwa zaproponowane w pracy algorytmy wydobywania reguł prototypowych cechują różne właściwości. Powoduje to, że w praktyce ich stosowania istotne jest sprawdzenie obydwu metod w celu wybrania lepszej z nich. Powyższą tezę można rozszerzyć o porównanie różnych algorytmów celem wybrania najwłaściwszego do analizowanego zagadnienia.

Rozdział 11

Podsumowanie

11.1 Wnioski

Przeprowadzona analiza zarówno empiryczna, jak i teoretyczna wskazuje na duże możliwości analizy danych w oparciu o metody reguł prototypowych. Analiza danych doświadczalnych wykazała, iż w większości wypadków uzyskane rezultaty były porównywalne lub nawet przewyższały wyniki uzyskane powszechnie uznanymi metodami ekstrakcji wiedzy - i to zarówno metod wydobywających reguły ostre, jak i rozmyte.
Uzyskane wyniki klasyfikacji rzeczywistych danych wskazują na bardzo dobre właściwości zaproponowanej w pracy adaptacji algorytmu warunkowego grupowania danych w połączeniu z algorytmem LVQ do klasyfikacji danych. Podobnie korzystnie wypada też porównanie metod doboru liczby prototypów. Algorytm wyścigu przy porównywalnych rezultatach znacznie przyspieszył uzyskanie odpowiedniej liczby prototypów przypadających na klasę w porównaniu z metodą wspinaczki.
Porównanie metod ekstrakcji reguł prototypowych progowych: algorytmu HDT z OPTDL udowadnia iż osiągają one podobne dokładności oraz zdolności generalizacji. Na korzyść algorytmu OPTDL przemawia jednak większa prostota i łatwość interpretacji uzyskanych wyników.
Opisana w rozdziale 9 równoważność reguł prototypowych oraz rozmytych pozwala na wzajemną transformację uzyskanych wyników pomiędzy obydwoma typami metod reprezentacji wiedzy. Dzięki przedstawionej analizie użytkownik ma możliwość interpretacji wyników bądź w postaci prototypów, bądź też w postaci reguł rozmytych. Kolejną korzyścią płynącą z równoważności obydwu systemów jest możliwość wzajemnej adaptacji algorytmów uczenia, tym bardziej że większość z opisanych w rozdziale 6 metod nie była dotychczas stosowana w problemie rozumienia danych.
Zaprezentowane w rozdziale 5 nowe probabilistyczne funkcje odległości dla atrybutów ciągłych typu HVDM, PVDM oraz LVDM pozwalają na poprawę dokładności klasyfikacji. Uzyskane w rozdziale 5.3.3 wyniki dowodzą, iż wzrost dokładności sięga często kilku procent. Zaproponowane w pracy odległości mają jednak wadę związaną z interpretacją uzyskanych wyników. Powoduje to, iż ich praktyczne wykorzystanie w procesie wydobywania reguł staje się ograniczone i dlatego też nie zostały one użyte do obliczeń opisanych w rozdziale 10.

11.2 Możliwości dalszych badań

Analiza budowy reguł prototypowych typu k-NNoraz PTR wskazuje, iż obydwa systemy posiadają różne właściwości. Systemy reguły typu PTR mają zasięg lokalny, gdyż pojedyncza reguła definiuje lokalną podprzestrzeń w przestrzeni danych wejściowych, podczas gdy reguły typu k-NNmają charakter globalny. Tym samym interesującym rozwiązaniem wydaje się być sekwencyjna integracja obydwu typów reguł w postaci jednolitej metody analizy danych. Rezultatem integracji powinien być algorytm, który w pierwszym kroku dokonywałby wstępnej klasyfikacji w oparciu o reguły typu k-NN, a następnie dla lokalnych skupisk wektorów klasyfikacja byłaby realizowana z wykorzystaniem reguł PTR.
Ciekawym obszarem badań wydaje się być analiza redukcji liczby prototypów zarówno dla reguł k-NN, jak i PTR. W literaturze mało jest przykładów realizacji tego zagadnienia w zastosowaniu do algorytmu RCE, co wydaje się być interesującym problemem. Podobnie problem selekcji wektorów referencyjnych dla algorytmu k-NNjest również otwarty. Dowodzą tego ciągle pojawiające się nowe publikacje z tej dziedziny.
Równie ważnym obszarem dalszych badań jest kwestia ważenia prototypów i wykorzystania do tego celu zasady maksymalizacji marginesu separowalności. Innym ciekawym obszarem zastosowań reguł prototypowych jest zagadnienie wydobywania reguł z nauczonych modeli o wiedzy typu *implicite*. Zagadnienie to jest obecnie rozwijane przez autora w zastosowaniu do wydobywania wiedzy z klasyfikatorów typu SVM, gdzie możliwe jest wykorzystanie algorytmów redukcji liczby wektorów wsparcia typu *metody zredukowanych zbiorów* (ang. reduced set methods) [149].

Rozdział 12

Dodatek 1

12.1 Opis zbiorów użytych w testach

12.1.1 Rak piersi

Zbiór danych dotyczących raka piersi uzyskano ze szpitali uniwersyteckich w Wisconsin (ang. wisconsin brest cancer). Zbiór zawiera dwie klasy opisujące typ nowotworu: złośliwy (ang. malignant) oraz łagodny (ang. bening). Zawierają one odpowiednio 241 (34,5%) oraz 458 (65,5%) przypadków opisanych poprzez 9 cech. Całość zbioru liczy 699 wektory, z czego 9 wektorów zawiera wartości brakujące. W testach porównawczych wektory zawierające wartości brakujące usunięto.

12.1.2 Wyrostek robaczkowy

Zbiór ten (ang. appendicitis) stanowi zbiór 8 testów medycznych przeprowadzonych dla 106 pacjentów podejrzanych o konieczność usunięcia wyrostka robaczkowego. Pacjentom tym przeprowadzono również biopsję, na podstawie której określono rzeczywistą konieczność operacji - podlegało jej 85 pacjentów. Ze zbioru danych udostępnionego przez Weissa [175] autor usunął jedną z cech ze względu na brakujące wartości. Zadaniem klasyfikacji jest przewidzenie na podstawie 7 testów medycznych konieczności dokonania operacji.

12.1.3 Cukrzyca

Cukrzyca (ang. pima indian diabetes) to zbiór danych pochodzący z repozytorium UCI [117] określający podejrzenie o cukrzycę wg. kryterium Światowej Organizacji Zdrowia. Każdy pacjent opisany jest 8 atrybutami porządkowymi bądź ciągłymi. Spośród badanych 768 pacjentów 500 sklasyfikowanych jest jako zdrowi, zaś 268 jako chorzy.

12.1.4 Sonar

Zbiór *sonar* został pobrany z repozytorium UCI i opisuje problem dyskryminacji sygnałów sonaru odbitych od przedmiotów metalowych o kształcie cylindrycznym oraz od skał. Baza ta składa się ze 111 wektorów opisujących sygnał pochodzący z odbicia

od metalowego przedmiotu cylindrycznego oraz 97 wektorów, których źródłem było echo skał. Każdy z wektorów opisany jest poprzez 60 zmiennych, z których każda odpowiada innej częstotliwości echa sygnału znormalizowanej do przedziału $(0 \ldots 1)$.

12.1.5 Choroby wątroby

Zbiór *choroby wątroby* (ang. BUPA liver disorders) pochodzi z repozytorium UCI i opisuje przypadki niewydolności wątroby. Całość zbioru stanowi 6 atrybutów oraz 345 przypadków. Poszczególne atrybuty to pięć różnych wartości badań krwi oraz jedna cecha opisująca ilość pitego w ciągu dnia alkoholu.

12.1.6 Irysy

Irysy jest zbiorem opisującym rozpoznawanie trzech gatunków irysów setosa, versicolor oraz virginica na podstawie czterech cech - długości oraz szerokości kielicha oraz długości i szerokości płatka. Zbiór *irysy* składa się ze 150 wektorów, po 50 wektorów dla każdego z gatunków. Zbiór ten jest bardzo prosty, gdyż rozkład wektorów w poszczególnych klasach dla cechy 3 i 4 posiada kształt Gaussowski.

12.1.7 Winorośl

Zbiór ten opisuje problem rozpoznawania trzech gatunków winorośli pochodzących z tego samego regionu Włoch. Podobnie jak *irysy*, zbiór ten jest prostym zbiorem często używanym do porównań. Składa się on z 13 atrybutów ciągłych opisujących różne parametry winorośli.

12.1.8 Jonosfera

Zbiór *jonosfera* (ang. ionosphere) stanowi ocenę możliwości radaru na podstawie jakości uzyskanych obrazów z jonosfery. Zbiór pochodzi z repozytorium UCI [117] i składa się z 351 wektorów opisanych przez 34 cechy ciągłe.

12.1.9 Lancet

Zbiór *lancet* (ang. lancet) stanowi opis 692 przypadków raka piersi, spośród których 235 jest złośliwych oraz 457 łagodnych. Każdy przypadek składa się z wieku pacjenta oraz 10 binarnych cech uzyskanych. Etykiety wektorów uzyskano na podstawie biopsji. Zbiór został udostępniony przez autorów Walkera, Crossa oraz Harrisona [171].

12.1.10 Choroby serca

Zbiór ten (ang. cleveland heart disease) stanowi opis pacjentów z chorobami serca. Składa się z 303 wektorów (każdy wektor opisuje jednego pacjenta) sklasyfikowanych początkowo w 5 kategoriach, a następnie przeetykietowanych do dwóch określających

osobę zdrową - 164 przypadki oraz chorą 139 przypadków. Zbiór oryginalnie składał się z 76 atrybutów, jednak autorzy wybrali 14 najbardziej istotnych cech, które powszechnie służą do testów porównawczych. Zbiór pochodzi z repozytorium UCI [117].

Bibliografia

[1] Adamczak R., Zastosowanie sieci neuronowych do klasyfikacji danych doświadczalnych, PhD THesis, Katedra Metod Komputerowych, Uniwersytet Mikołaja Kopernika, 2001

[2] Aha D. W., Kibler D., Albert M.K., Instance-Based Learning Algorithms, Kluwer Academic Publishers, Machine Learning, vol. 6, pp. 37-66, 1991

[3] Aha D., Feature Weighting for Lazy Learning Algorithms, In Feature Extraction, Construction, and Selection. A Data Mining Perspective, Huan, Liu and Hiroshi, Motoda (eds.)., Kluwer Academic Publisher, pp 13-32, 1998

[4] Aha D., Tolerating noisy, irrelevant and novel attributes in instancebased learning algorithms, International Journal of Man-Machine Studies, vol. 36, pp. 267-287, 1992

[5] Baffes P. T., Mooney R.J., Symbolic revision of theories with M-of-N rules. Proc. of IJCAI, Chambery, France, pp. 1135-1140, 1993

[6] Bereta M., Hybrid immune algorithm for feature selection and classification of ECG signals, AI-METH Series, Gliwice, 2005

[7] Bermejo Sánchez, Learning with nearest neighbour classifiers, PhD Thesis, Technical University of Catalonia, 2000

[8] Bezdek J., Kuncheva L., Some Notes on Twenty One (21) Nearest Prototype Classifiers, Proc. of the Joint International Workshops on Advances in Pattern Recognition, Lecture Notes In Computer Science, vol. 1876, pp. 1-6, 2000

[9] Bezdek J.C., Kuncheva L.I., Nearest prototype classifier designs: An experimental study, International Journal of Intelligent Systems, vol. 16, pp. 1445-1473, 2001

[10] Bhattacharya B.K., Toussaint G.T., Poulsen R.S., Application of proximity graphs to editing nearest neighbor decision rules. SOCS, 1992

[11] Biesiada J., Duch W., Feature Selection for High-Dimensional Data: A Kolmogorov-Smirnov Correlation-Based Filter Solution, Proc. of 4th International Conference on Computer Recognition Systems, CORES 2005, Rydzyna, Advances in Soft Computing, Computer Recognition Systems, pp. 95-105, 2005

[12] Biesiada J., Fsel++,
http://metet.polsl.katowice.pl/ jbiesiada/__strona_infosel/

[13] Bilgic T., Turksen B., Measurement of Membership Functions Theoretical and Empirical Work Measurements of Membership Functions: Theoretical and Empirical Work, in Dubois D. and Prade (eds.) H., vol. 1, Fundamentals of Fuzzy Sets. Kluwer, pp. 195–232, 2000

[14] Bishop C., Pattern Recognition and Machine Learning, Springer, 2006

[15] Blachnik M, Duch W, Wieczorek T., Threshold rules decision list, in Methods of Artificial Intelligence, AI-METH Series, Gliwice, 2005

[16] Blachnik M, Duch W, Wieczorek T, "Probabilistic distance measures for prototype-based rules. Proc. of ICONIP,"Taipei, Taiwan, pp. 445-450, 2005

[17] Blachnik M, Duch W, Wieczorek T, Selection of prototypes rules – context searching via clustering. Proc. of International Conference on Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science, vol. 4029, Springer Verlag, Poland, 2006

[18] Blachnik M. Warunkowe metody rozmytego grupowania w zastosowaniu do uczenia radialnych sieci neuronowych, Master thesis, Silesian University of Technology, Gliwice, Poland, 2002

[19] Blachnik M., Duch W. Prototype-based threshold rules, Proc. of ICONIP, Springer, Lecture Notes in Computer Science, vol. 4234, 2006

[20] Blanzieri E., Ricci F., Probability Based Metrics for Nearest Neighbor Classification and Case Based Reasoning, Proc. 3rd Int. Conf. on Case-Based Reasoning, 1999

[21] Blanzieri E., Ricci F., Metric for Nearest Neighbor Classification, Proc. 16th International Conf. on Machine Learning, 1999

[22] Breiman L., Friedman J.H., Oslhen R.A., Stone C.J., Classification and Regression Trees. Belmont, CA: Wadsworth International Group, 1984.

[23] Cameron-Jones R., Instance Selection by Encoding Length Heuristic with Random Mutation Hill Climbing. In Proc. of the Eighth Australian Joint Conference on Artificial Intelligence, pp. 99-106, 1995

[24] Cataron A., Andonie R. Energy Generalized LVQ with Relevance Factors. Proc. IJCNN, 2004

[25] Chang, Chin-Liang. Finding Prototypes for Nearest Neighbor Classifiers. IEEE Transactions on Computers, vol. 23(11), pp. 1179-1184, 1974

[26] Chi J., Entropy based feature evaluation and selection technique, Proc. of 4-th Australian Conf. on Neural Networks (ACNN'93), pp. 181-196, 1993.

[27] Chopra S., Hadsell R., LeCun Y., Learning a Similarity Metric Discriminatively, with Application to Face Verification, Proc. of CVPR'05, vol. 1, 2005

[28] Cichosz P., Systemy uczące się, Wydawnictwa Naukowo-Techniczne, Warszawa, 2000

[29] Clark P., Niblett T., The CN2 Induction Algorithm, Machine Learning Jurnal, 3(4), pp.261-283, 1989

[30] Clark P., Rue Induction with CN2: Some Recent Improvements, Machine Learning, Procc. of Fifth European Connerence EQSL-91, Springer Verlag, Berlin, pp. 151-163, 1991

[31] Cordn O., del Jesus M.J., Herrera F., A proposal on reasoning methods in fuzzy rule-based classification systems, International Journal of Approximate Reasoning, vol.20 (1), pp. 21-45, 1999

[32] Cost S., Salzberg S., A weighted nearest neighbor algorithm for learning with symbolic features, Machine Learning, vol. 10(1), 57-78, 1993

[33] Crammer K., Gilad-Bachrach R, Navot A., Margin Analysis of the LVQ Algorithm, NIPS'02, 2002

[34] Craven M. W., Shavlik J. W., Extracting tree-structured representations of trained networks, Advances in Neural Information Processing Systems, D. Touretzky, M. Mozer, and M. Hasselmo, Eds. Cambridge, MIT Press, vol. 8, pp. 24–30, 1996

[35] Czogała E., Łęski J., Fuzzy and neuro-fuzzy intelligent systems, Physica-Verlag, Springer-Verlag Com., Heidelberg, New York, 2000

[36] Datta P., Kibler D., Learning Symbolic Prototypes, Proc. 14th International Conference on Machine Learning, 1997

[37] Datta P., Kibler D., Symbolic Nearest Mean Classifiers, AAAI/IAAI, pp. 82-87, 1997

[38] Domeniconi C., Peng J., Gunopulos D., Locally Adaptive Metric Nearest-Neighbur Classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24(9), pp 1281-1285, 2002

[39] Domingos P., Rule Induction and Instace Based Learning: A Unified Approach. Proc. fo the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Canada, Morgan Kaufmann, pp. 1226-1232, 1995

[40] Drummond C., Holte R., What ROC Curves Can't Do (and Cost Curves Can), Proc of the ROC Analysis in Artificial Intelligence, 1st International Workshop, Valencia, Spain, pp. 19-26, 2004

[41] Dubisson M. P., Jain A.K., Modified hausdorff distance for object matching. Preceedings of the International Conference on Pattern Recognition, vol.1, pp. 566-568, 1994

[42] Duch W, Blachnik M, Fuzzy rule-based systems derived from similarity to prototypes. Proc. of ICONIP, Lecture Notes in Computer Science, vol. 3316, 2004

[43] Duch W, Wieczorek T, Biesiada J, Blachnik M, Comparison of feature ranking methods based on information entropy. Proc. of International Joined Conference on Neural Networks, Budapest, IEEE Press, 2004

[44] Duch W, Adamczak R, Diercksen G.H.F., Neural Networks from Similarity Based Perspective. New Frontiers in Computational Intelligence and its Applications. Ed. M. Mohammadian, IOS Press, Amsterdam, pp. 93-108, 2000

[45] Duch W, Adamczak R, Diercksen G.H.F. Classification, Association and Pattern Completion using Neural Similarity Based Methods. Applied Mathematics and Computer Science , vol. 10(4), pp. 101-120, 2000

[46] Duch W, Itert L, Committees of Undemocratic Competent Models. International Conference on Artificial Neural Networks (ICANN) and International Conference on Neural Information Processing (ICONIP), Istanbul, pp. 33-36, 2003

[47] Duch W, Grudziński K and Stawski G, Symbolic features in neural networks. 5th Conference on Neural Networks and Soft Computing, Zakopane, pp. 180-185, 2000

[48] Duch W., Adamczaka R., Diercksen G.H.F., Distance-based Multilayer Perceptrons, Computational Intelligence for Modelling Control and Automation. Neural Networks and Advanced Control Strategies. Ed. M. Mohammadian, IOS Press, Amsterdam, pp. 75-80, 1999

[49] Duch W., Similarity based methods: a general framework for classification, approximation and association, Control and Cybernetics, vol. 29, pp. 937-968, 2000

[50] Duch W., Jankowski N., Survey of neural transfer functions. Neural Computing Surveys, vol. 2, 163–212, 1999

[51] Duch W., Grudziński K. Prototype based rules - a new way to understand the data. Proc. of IJCNN'01, Washington D.C., USA, pp. 1858-1863, 2001

[52] Duch W., Grudziński K., Meta-learning via search combined with parameter optimization. Intelligent Information Systems, Advances in Soft Computing, Physica Verlag, Springe, pp. 13-22, 2002

[53] Duch W., Grudzinski K., The Weighted k–NN with Selection of Features and Its Neural Realization. th Conference on Neural Networks and Their Applications, Zakopane, pp. 191-196, 1999

[54] Duch W., Adamczak R., Feature Space Mapping Network for Classification, Proc. second Conference on Neural Networks and Their Applications, Orle Gniazdo, Poland, vol. 1, pp. 125-130, 1996

[55] Duch W., Adamczak R., Grąbczewski K., A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 12, pp. 277-306, 2001

[56] Duch W., Setiono R., Żurada J. Computational Intelligence Methods for Rule-Based Data Understanding, Proceedings of the IEEE, vol. 92(5), pp. 771-805, 2004

[57] Duch W., Adamczak R., Grabczewski K., Extraction of logical rules from training data using backpropagation networks., Proc. 1st Online Workshop Soft Computing [Online], pp. 25–30.

[58] Duch W., Wieczorek T., Biesiada J., Blachnik M. Comparision of feature ranking methods based on information entropy. Proc. of International Joint Conference on Neural Networks (IJCNN'04), Budapest, IEEE Press, pp. 1415-1420, 2004

[59] Duch W., Diercksen G., Feature Space Mapping as a universal adaptive system, Computer Physics Communications, vol. 87, pp. 341-371, 1995

[60] Duda R.O., Hart P.E., Stork D.G., Pattern Classification and Scene Analisys, New York: John Wiley & Sons, 1973.

[61] Duda R.O., Hart P.E., Stork D.G., Pattern Classification, New York: John Wiley & Sons, 2nd ed, 2001.

[62] Dudani S.A., The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, SMC, 6(4), pp 325–327, 1976

[63] Feng G., A Comprehensive Overview of Basic Clustering Algorithms, June 22, 2001

[64] Ghosh A., Biehl M, Freking A, Reents G., A theoretical framework for analyzing the dynamics of learning vector quantization: A statistical physics approach, Mathematics and Computing Science, University Groningen, 2004

[65] Grąbczewski K., Duch W., The separability of split value criterion. 5th Conference on Neural Network and Soft Computing, Polish Neural Network Society, Zakopane, Poland, pp. 201-208, 2000

[66] Grąbczewski K., Duch W., Heterogenous forests of decision trees. Springer Lecture Notes in Comp. Science, vol. 2415, pp. 504-509, 2002

[67] Grochowski M., Jankowski N.: Comparison of Instance Selection Algorithms II. Results and Comments, Lecture Notes in Artificial Intelligence, vol. 3070, pp. 580-585, 2004.

[68] Guyon I., Gunn S., Nikravesh M., Zadeh L., Feature Extraction, Foundations and Applications, Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2006.

[69] Hall M.A., Correlation based feature selection for machine learning, PhD thesis, Dept. of Comp. Science, Univ. of Waikato, Hamilton, New Zealand, 1998

[70] Hand D., Mannila H., Smyth P., Principles of Data Mining, MIT Press , 2001

[71] Hart P.E., The condensed nearest neighbor rule. IEEE Trans. on Information Theory, vol. 16, pp. 515-516, 1968

[72] Henry Brighton, Chris Mellish. Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery, vol. 6, pp.153–172, 2002

[73] Höppner F., Klawonn F., Kruse R. Runkler T., Fuzzy Cluster Analysis, Wiley, 1999

[74] Hullermeier E., Dubios D., Prade H., Fuzzy Rules in Case-Based Reasoning, 1999

[75] Hullermeier E., Possibilistic Instance-Based Learning, Artificial Intelligence, vol 148, Issues 1-2, pp. 335-383, 2003

[76] Ichihashi H., Shirai T., Nagasaka K., Miyoshi T., Neuro-fuzzy ID3: a method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning, Fuzzy Sets and Systems, 81 pp.157-167, 1996

[77] IJCNN Challenge 2007

[78] Jacobs D.W., Weinshall D., Gdalayahu Y., Classification with non-metric distances: image retrival and class representation. IEEE Trans. Pattern Anal. Mach. Intell, 22(6), pp. 583-600, 2000

[79] Jang R., Sun C. T., Functional Equivalence between Radial Basis Functions Networks and Fuzzy Inference Systems, IEEE Transactions on Neural Networks, 1996

[80] Jang R., ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE Trans. on Systems, Man, and Cybernetics, vol.23, pp. 665-684, 1993

[81] Jang R.J., Sun C.T., Mizutani E., Neuro Fuzzy and Soft Computing. A computational Approach to Learning and Machine Intelligence, Pronce Hall, Matlab Curriculum Series, 1996

[82] Jankowski N., Grochowski M. Comparison of Instance Selection Algorithms I. Algorithms Survey, Lecture Notes in Artificial Intelligence, vol. 3070, pp. 598-603, 2004.

[83] Jankowski N., M. Grochowski. Instances selection algorithms in the conjunction with LVQ. In M. H. Hamza, Artificial Intelligence and Applications, Innsbruck, Austria, ACTA Press, pp. 453-209, 2005

[84] Jankowski N., Grąbczewski K., Duch W., GhostMiner 3.0., FQS Poland, Fujitsu, Kraków, Poland, 2004.

[85] Karayiannis N.B., A Methodology for Constructing Fuzzy Algorithms for Learning Vector Quantization., IEEE Transactions on Neural Networks, vol. 8(3), pp. 505-518, 1997

[86] Karayiannis N.B., An Axiomatic Approach to Soft Learning Vector Quantization and Clustering IEEE Transactions on Neural Networks, VOL. 10(5), 1999

[87] Karayiannis N.B., Soft Learning Vector Quantization and Clustering Algorithms Based on Non-Euclidean Norms: Multinorm Algorithms, IEEE Transactions on Neural Networks, vol. 14(1), 2003

[88] Karayiannis N.B., An Integrated Approach to Fuzzy Learning Vector Quantization and Fuzzy -Means Clustering, IEEE Transactions On Fuzzy Systems, vol. 5(4), NOVEMBER 1997

[89] Kasabov N., Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, MIT Press, Cambridge, Massachusetts London, England, 1996

[90] Kasabov N., Evolving Fuzzy Neural Network for Supervised Unsupervised On-line, Knowledge-based Learning, IEEE Trans. on Man, Machine and Cybernetics, 2001

[91] Kasif S., Salzberg S., Waltz D., Rachlin J., Aha D., Towards a Framework for Memory Based Reasoning Submitted for publication. Copy found online, 1995

[92] KEEL: Knowledge Extraction based on Evolutionary Learning, http://sci2s.ugr.es/keel/

[93] Kim S.W., Oommen J. Recursive Prototype Reduction Schemes Applicable for Large Data Sets, Proc. of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes In Computer Science, vol. 2396, pp. 528-537, 2002

[94] Kim Sang-Woon, Oommen B. J., Recursive Prototype Reduction Schemes Applicable for Large Data Sets, Lecture Notes In Computer Science, vol. 2396, pp 528 - 537, 2002

[95] Kohavi R., John G.H., Wrappers Approach, in Feature Extraction, Construction and Selection: A Data Mining Perspective, Liu H., Motoda H. edt., Springer, 1998

[96] Kohavi R., John G., Wrappers for Feature Subset Selection. Artificial Intelligence, special issue on relevance, vol. 97, N. 1-2, pp. 273-324, 1997

[97] Kohavi R., The Power of Decision Table, European Conference on Machine Learning, 1995

[98] Kohonen T., Self-Organizing Maps, Springer-Verlag, 2001.

[99] Kordos M., Duch W., Search-based Training for Logical Rule Extraction by Multilayer Perceptron, Proc. of the ICANN/ICONIP, Istanbul, pp. 86-89, 2003

[100] Kuncheva L., On the equivalence between fuzzy and statistical classifiers, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15, pp. 245-253, 1996

[101] Kuncheva L., Fuzzy Classifier Design, Studies in Fuzziness and Soft Computing, Physica-Verlag, 2000

[102] Kuncheva L., Genetic algorithms for feature selection for parallel classifiers. Information Processing Letters, pp. 163-168, 1993

[103] Kuncheva L.I., Bezdek J.C., An Integrated Framework for Generalized Nearest Prototype Classifier Design, International Jurnal of Uncertainty, Fuzziness and Knowledge Systems, vol. 6(5), pp.437-457, 1998

[104] Kuncheva L.I., Lakow D., RBF Networks Versus Fuzzy If-Then Rules for Classification, Int. J. Knowledge-Based Intelligent Eng. Systems, vol. 2, pp. 203-210, 1998

[105] Kuncheva L.I., How good are fuzzy if-then classifiers? IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 30(4), 501-509, 2000

[106] Kuncheva L.I., Bezdek J.C., Presupervised and postsupervised prototype classifier design, IEEE Transactions on Neural Networks, vol. 10(5), 1142-1152, 1999

[107] Kuncheva L.I., Bezdek J.C., Nearest prototype classification: Clustering, genetic algorithms or random search? IEEE Transactions on Systems, Man, and Cybernetics, vol.C28(1), 160-164, 1998

[108] Kuncheva L.I., Jain L.C., Nearest neighbor classifier: Simultaneous editing and feature selection.Pattern Recognition Letters, vol. 20, pp. 1149-1156, 1999

[109] Lebanon G., Metric Learning for Text Documents, Proc. of TPAMI, 2006

[110] Lewis DD, Yang X, Rose T, Li F., RCV1: A new benchmark collection for text categorization research. JMLR, vol. 5, pp. 361-97, 2004.

[111] Lim H.S., Improving kNN Based Text Classification with Well Estimeted Parameters, Precedings of ICONIP, Lecture Notes in Computer Science, 3316 pp. 516-523, 2004

[112] Liu H., Hussain F., Tan C. L., Dash M. Discretization: An Enabling Technique. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, vol. 6, pp. 393–423, 2002

[113] Lopez de Mantaras R., A Distance-Based Attribute Selecting Measure for Decision Tree Induction, Machine Learning vol. 6, pp. 81-92, 1991.

[114] Łęski J., A new generalized weighted conditional fuzzy clustering", BUSEFAL, vol.81, pp.8–16, 2000

[115] Łęski J., Ordered weighted generalized conditional possibilistic clustering, in J.Chojcan, J. Łęki (Eds.), Zbiory rozmyte i ich zastosowania, Wydawnictwa Politechniki Śląskiej, Gliwice, pp. 469–479, 2001

[116] Maron O., Moore A.: The Racing Algorithm: Model Selection for Lazy Learners. Artificial Intelligence Review, vol. 11, 193-225, 1997.

[117] Mertz C.J., Murphy P.M., UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-databases

[118] Michalksi R.S., Mozetic I., Hong J., Lavrac N., The AQ15 inductive learning system: an overview and experiments. In Procc of IMAL 1986, Orsay, France, 1986

[119] Michalski R.S., Larson J., Incremental generation of vl_1 hypotheses: the underlying methodology and description of program aq11. ISG 83-5, Dept. ofComputer Science, Univ. ofIllinois at Urbana-Champaign, Urbana, 1983.

[120] Mitchell T.M., Machine Learning. McGraw-Hill, 1997

[121] Mollineda R., Ferri F., Vidal E., An efficient prototype merging strategy for the condensed 1-nn. Pattern Recognition, 2002

[122] Morring B., Martinez T., Weighted Instance Typicality Search (WITS): A Nearest Neighbor Data Reduction Algorithm, Intelligent Data Analysis, vol 8(1), pp. 61-78, 2004

[123] Müller K., Mika S., Rätsch G., Tsuda K, Schölkopf B., An Introduction to Kernel-Based Learning Algorithms, IEEE Transactions on Neural Networks, vol. 12(2), 2001

[124] N. Jankowski, Ontogeniczne sieci nauronowe, O sieciach zmieniających swoją strukturę, AOW Exit, Warszawa, 2003

[125] Nauck D., Klawonn F., Kruse R.: Foundations on Neuro-Fuzzy Systems. Wiley, Chichester, 1997.

[126] Nauck D.D., Data Analysis with Neuro-Fuzzy Methods, Habilitation thesis, Otto-von-Guericke-Universit, Magdeburg, 2000

[127] Nauck, D., Design and Implementation of Neuro-Fuzzy Data Analysis Tool in Java. Technische Universitat Brauschweig, Brauschweig, 1999.

[128] NIPS Challenge 2003

[129] Nowak E., Jurie F., Learning Visual Distance Function for Object Identification from one Example, NIPS'06, 2006

[130] Núñez H., Angulo C., Catalŕ A., Rule extraction from support vector machines, Proc. of ESANN'2002, Bruges, Belgium, 2002

[131] Odorico R., Learning Vector Quantization with Training Count (LVQTC), Neural Networks, 10(6), pp. 1083-1088, 1997

[132] Oh I.S., Lee J.S., Moon B.R. Hybrid Genetic Algorithms for Feature Selection. Pattern Analysis and Machine Intelligence, 26(11), pp. 1424-1437, 2004

[133] Ossowski S., Sieci Neuronowe w Ujęciu Algorytmicznym, WNT, Warszawa, 1996

[134] Pedrycz W., Conditional Fuzzy Clustring in the Design of Radial Basis Function Neural Networks, IEEE Trans NN, vol. 9(4), 1998

[135] Pedrycz W., Conditional Fuzzy C-Means, Pattern Recognition Letters, vol. 17, 625-632, 1996.

[136] Pedrycz W., Fuzzy set technology in knowledge discovery, Fuzzy Sets and Systems 98, 279-290, 1998

[137] Pękalska E., Duin R., Paclik P., Prototype selection for dissimilarity-based classifiers, Pattern Recognition 39(2), 189-208, 2006

[138] Piegat A., Modelowanie i sterowanie rozmyte, AOW Exit, Warszawa, 2003

[139] Plaza E., Aamodt A., Case-based Reasoning-Fundamental Issues, Methodological Variations, and System Approaches. AICOM, vol. 7(1), pp. 39-59, 1994

[140] Pothos E.M., The Rules versus Similarity distinction, Behavioral and Brain Sciences, Cambridge University Press, 2004

[141] Provost F., Fawcett T., Kohavi R., The Case Against Accuracy Estimation for Comparing Induction Algorithms

[142] Pudil P., Novovivcov'a J., Kittler J., Floating search methods in feature selection. Pattern Recognition Letters, v.15, pp. 1119-1125, 1994

[143] Quinlan J.R., C4.5: Programs for machine learning, Morgan Kaufman, CA, 1993

[144] Rutkowska D., Sieci rozmyto-neuronowe do klasyfikacji: architektura typu RBF i MLP - system NEFCLASS, Automatyka, v.9/3 pp. 689-698, 2005

[145] Rutkowska D., Rutkowski L., Systemy rozmyte i rozmyto-neuronowe, Biocybernetyka i Inżynieria Biomedyczna 2000, tom. 6, Sieci Neuronowe, AOW PLJ, 1999

[146] Rutkowski L., Kacprzyk J., Neural Networks and Soft Computing, Physica-Verlag, Springer-Verlag, 2003

[147] Rutkowski L., Metody i Techniki Sztucznej Inteligencji, Wydawnictwo Naukowe PWN, 2005

[148] Schmidt R., Pollwein B, Gierl L., Experiences with Case-Based Reasoning Methods and Prototypes for Medical Knowledge-Based Systems. Proc. AIMDM'99, Aalborg, Denmark, Lecture Notes in Computer Science, vol. 1620, 1999

[149] Schölkopf B., Smola A. Learning with Kernels, MIT Press, Cambridge, MA, 2002

[150] Schwenker F., Kestler H., Palm G., Three learning phases for radial-basis-function networks, Neural Networks 14, 439-458, 2001

[151] Schwenker F., Kestler H., Palm G., Three learning phases for radial-basis-function networks, Neural Networks 14, pp. 439-458, 2001

[152] Seeger M., Bayesian model selection for Support Vector machines, Gaussian processes and other kernel classifiers, NIPS, 2001

[153] Setion R., Liu H., Improving backpropagation learning with feature selection. Applied Intelligence: The International Journal of Artifical Intelligence, Neural Networks, and Complex Problem-Solving Technologies, vol. 6, pp. 129-139, 1996.

[154] Setiono R., Extracting M-of-N rules from trained neural networks, IEEE Transactions on Neural Networks, vol. 11, pp. 306-512, 2001

[155] Setiono R., Liu H. Symbolic Representation of Neural Networks. IEEE Computer, vol.29, 3 pp. 71-77, 1996

[156] Shanonn C.E., Weaver W., The Mathematical Theory of Communication. Urbana, IL, University of Illinois Press, 1946.

[157] Short R. D., Fukunaga K., A new nearest neighbour distance measure. In Proceedings of the 5th IEEE International Conference on Patter Recognition, Miami beach, FL, pp 81-86, 1980

[158] Shridhar D.V., Bartlett E.B., Seagrave R.C., Information theoretic subset selection. Computers in Chemical Engineering, vol. 22, pp. 613-626, 1998.

[159] Siedlecki, W. and Sklansky, J. A Note on Genetic Algorithms for Large-Scale Feature Selection. PRL, vol. 10, pp. 335-347, 1989

[160] Skalak D. B., Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms, International Conference on Machine Learning, Morgan Kaufmann, pp. 293-301, 1994

[161] Smyth P., Clustering using Monte Carlo Cross-Validation, "Knowledge Discovery and Data Mining,"pp. 126-133, 1996

[162] Stanfill C. Waltz D., Toward memory-based reasoning. Communications of the ACM, 29(12), pp 1213-1228, 1986

[163] Stanfill C., Waltz D., Toward memory-based reasoning. Communications of the ACM, 29(12), pp 1213-1228, 1986

[164] Stąpor K., Automatyczna Klasyfikacja Obiektów, Exit, Warszawa, 2005

[165] Thrun S., Extracting rules from artifcial neural networks with distributed representations, Advances in Neural Information Processing Systems 7, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MIT Press, 1995

[166] Tomek I., An experiment with the edited nearest-neighbor rule. IEEE Trans. on Systems, Man, and Cybernetics, 6 pp. 448-452, 1976

[167] Tsang I. W., Kwok J. T., Distance Metric Learning with Kernels, Proc. of ICANN, 2003

[168] Ulusoy, I., C. M. Bishop, Generative versus discriminative models for object recognition. In Proc. IEEE International Conference on Computer Vision and Pattern Recognition, CVPR., San Diego, 2005.

[169] van Rijsbergen CJ., Foundation of evaluation. Journal of Documentation 30:4, pp. 365-73, 1974

[170] Vapnik V., The Nature of Statistical Learning Theory 2nd edt., Springer, 2000

[171] Walker A.J., Cross S.S., Harrison R.F., Visualization of biomedical datasets by use of growing cell structure networks: a novel diagnostic classification technique. Lancet, vol. 354, pp. 1518-1522, 1999.

[172] WCCI Challenge 2006

[173] Webb A., Statistical Pattern Recognition, John Wiley and Sons Ltd., 2002

[174] Weinberger K. Q., Blitzer J., Saul L. K., Distance Metric Learning for Large Margin Nearest Neighbor Classification, in Proceedings of the 19th annual conference on Neural Information Processing Systems 2005, Vancouver CA., 2005

[175] Weiss S. M. , Kapouleas I., An empirical comparison of pattern recognition, neural nets and machine learning classification methods. J. W. Shavlik, T. G. Dietterich, redaktorzy, Readings in Machine Learning. Morgan Kauffman Publ. CA, 1990.

[176] Wess S., Altho K.D., Derwand G. Using k-d trees to improve the retrieval step in case-based reasoning, editors, Topics in Case-Based Reasoning, pp 167–181. SpringerVerlag, Berlin, 1994.

[177] Wettschereck, Dietrich, Aha D. W., Mohri T., A Review and Comparative Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review, vol. 11, issues 1-5, Special Issue on Lazy Learning, Kluwer Academic Publishers, pp. 273-314, 1997

[178] Wieczorek T, Blachnik M, "Duch W. Heterogeneous distance functions for prototype rules: influence of parameters on probability estimation. International Journal of Artificial Intelligence Studies (w druku)"

[179] Wieczorek T., Intelligent control of the electric-arc steelmaking process using artificial neural networks. Comp. Methods in Material Science, vol.6, No. 1, 2006

[180] Wieczorek T., Blachnik M., Duch W., "Influence of probability estimation parameters on stability of accuracy in prototype rules using heterogeneous distance functions,"Proc. of Artificial Intelligence Studies, vol. 2, Siedlce, 2005.

[181] Wilson D. R., Martinez T.R., Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research 6, 1997

[182] Wilson D. R., Martinez T.R., An Integrated Instance-Based Learning Algorithm, Computational Intelligence, 16(1), pp. 1-28, 2000

[183] Wilson D.L., Assymptotic properties of nearest neighbour rules using edited data., IEEE Trans. on Systems, Man, and Cybernetics, SMC-2, pp. 408-421, 1972

[184] Wilson D.R., Martinez T.R., Reduction techniques for instance-based learning algorithms. Machine Learning vol. 38, pp. 257-268, 2000

[185] Wilson D.R., Martinez T.R., Instance Pruning Techniques. In Fisher, D., ed. Machine Learning: Proc. of the Fourteen International Conference (ICML'97), Morgan Kaufmann Publishers, San Francisko, CA , pp. 408-421, 1977

[186] Witten I., Frank E., Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[187] Zhou, Q., Purvis, M., and Kasabov, N., A Membership Function Selection Method for Fuzzy Neural Networks, Progress in Connectionist-Based Information Systems, Proc. of the 1997 ICONIP ANZIIS ANNES'97, Springer, Singapore, pp. 785-788, 1998

[188] Zien A., Rätsch G., Mika S., Schölkopf B., Lengauer T., Müller K.-R., Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics, 16(9):799–807, 2000.