Instytut Badań Systemowych
Polska Akademia Nauk

Krzysztof Grąbczewski
kgrabcze@phys.uni.torun.pl

Zastosowanie kryterium separowalności do generowania reguł klasyfikacji na podstawie baz danych

Rozprawa doktorska
pod kierunkiem prof. dra hab. Włodzisława Ducha

Warszawa, 2003

Streszczenie

Praca podejmuje problemy odkrywania wiedzy w bazach danych. Koncentruje się na rozwiązywaniu zagadnień klasyfikacji obiektów, poprzez tworzenie (za pomocą adaptacji parametrów) modeli, które pozwalają jednocześnie na uzyskiwanie wysokiej poprawności klasyfikacji oraz na przedstawienie zrozumiałego opisu modelu. Uwaga skupiona została na modelach, które można jednoznacznie opisać w języku logiki klasycznej.

Rozprawa szczegółowo przedstawia opracowane przez jej autora kryterium separowalności, poddaje wnikliwej analizie jego własności oraz prezentuje szereg zastosowań. Pokazuje jak tworzyć drzewa decyzji i opisywać ich działanie regułami logicznymi, jak budować heterogeniczne drzewa decyzji, a także zestawy alternatywnych drzew homo- i heterogenicznych (lasy).

Praca omawia oparte na kryterium separowalności metody pozwalające na dyskretyzację danych numerycznych, uciąglanie danych symbolicznych, oraz wykorzystanie kryterium separowalności do selekcji cech. Metody te mogą wspomagać inne systemy klasyfikacji, stając się w ten sposób częściami złożonych modeli heterogenicznych o dużej wszechstronności.

Oprócz technik opartych na kryterium separowalności, znalazły tu miejsce również metody generowania reguł logicznych wykorzystujące zalety sztucznych sieci neuronowych.

Wiele prezentowanych idei zostało zapisanych w postaci algorytmów. Wszystkie te algorytmy zostały zaimplementowane i przetestowane. Praca przedstawia i komentuje rezultaty wykonanych testów. Wyniki zostały przedstawione na tle aktualnych osiągnięć nauki w tej dziedzinie, porównane z rezultatami otrzymanymi innymi metodami, znanymi z literatury.

Rozprawa dyskutuje także metody porównywania wyników, sposoby postępowania w przypadku niepełności danych, podejścia do szacowania prawdopodobieństw przynależności do klas, wynikających z drzew decyzji i reguł klasyfikacji i inne zagadnienia związane z wydobywaniem wiedzy z baz danych.

Podziękowania

Pragnę wyrazić swoją wdzięczność promotorowi mojej pracy prof. Włodzisławowi Duchowi, a także kolegom z Katedry, za wiele inspirujących, żywiołowych dyskusji oraz za konstruktywną krytykę.

Z całego serca dziękuję również moim najbliższym i przyjaciołom za nieocenione wsparcie, którym mnie nieustannie wzmacniali, w szczególności mojej żonie, Beacie, która dzielnie znosiła wszystkie ciężary, jakie na nią spadły zwłaszcza w końcowym etapie moich prac nad rozprawą.

Spis treści

Spis rysunków

Spis tabel

Spis algorytmów

Rozdział 1

Wstęp

Rozwój elektroniki i coraz szersze zastosowania komputerów powodują, że na nośnikach magnetycznych, optycznych i magnetooptycznych gromadzone są ogromne ilości różnego rodzaju danych. Dane nie mają żadnej wartości, dopóki nie mamy możliwości wydobycia z nich informacji. W przypadku olbrzymich baz danych, bądź takiej ich postaci, która nie jest dla człowieka w naturalny sposób czytelna, skorzystanie z wiedzy, która się tam kryje, wymaga zastosowania narzędzi potrafiących dokonać stosownej selekcji – oddzielenia tego co ważne (z pewnego punktu widzenia) od tego co jest tylko szumem.

W ostatnich dziesięcioleciach znacznie rozwinęła się dziedzina nauki, którą nazywa się *Inteligencją Obliczeniową* (ang. *Computational Intelligence* – CI). Pewna jej część zajmuje się systemami klasyfikacji obiektów, które znajdują zastosowania nie tylko w diagnostyce medycznej, ale także w bankowości, finansach, bioinformatyce i wielu innych dziedzinach.

Problem odkrywania wiedzy w różnego rodzaju bazach danych, nazywany również dogłębną analizą, eksploracją lub drążeniem danych (ang. *data mining*), podejmowany był na wiele różnych sposobów i przez liczne ośrodki naukowe zajmujące się sztuczną inteligencją. Często spotyka się podejścia na tyle oryginalne (np. bardzo wąsko specjalizowane), że bardzo trudno jest rzetelnie ocenić ich rzeczywiste możliwości, gdyż nie można porównać otrzymanych za pomocą tych metod wyników z innymi.

Inaczej jest jednak w dziedzinie klasyfikacji obiektów, która oferuje sprawne metody weryfikacji jakości systemów. Nie oznacza to, że można wskazać jeden uniwersalny sposób oceny klasyfikatorów – istnieje wiele różnych celów jakim mają one służyć, a więc i kryteria ich oceny muszą być różne. Podstawowym wymogiem jest przypisywanie klas nieznanym obiektom z jak największą poprawnością, ale często w cenie jest również umiejętność uzasadnienia podejmowanej decyzji.

Przedstawianie zdobytej informacji w sposób przystępny dla człowieka nie

tylko zwiększa zaufanie do diagnoz systemu (poprzez umożliwianie weryfikacji ich podstaw), ale także pozwala na odkrycie tej części wiedzy ekspertów, która miała największy wpływ na podejmowane decyzje pomimo tego, że często sami eksperci nie są w stanie przedstawić podobnego uzasadnienia swojej diagnozy.

Odkrywanie wiedzy w danych dla problemów klasyfikacji może też polegać na wyszukiwaniu tych cech, które najlepiej odróżniają od siebie różne klasy. Na przykład w medycynie bardzo istotnym jest (zarówno z punktu widzenia czasu dochodzenia do właściwej diagnozy jak i koniecznych do poniesienia nakładów finansowych) kierowanie pacjentów na te badania, które potrafią jak najszybciej, najtrafniej, najtaniej i możliwie bezinwazyjnie doprowadzić do właściwej diagnozy.

Wspomaganie decyzji ekspertów przez systemy inteligencji obliczeniowej jest nieuniknionym kierunkiem rozwoju, bo przez odpowiednie stosowanie, pomoże uniknąć wielu kosztów zawodności ludzkiej ekspertyzy.

Praca przedstawia szereg metod, które służą dogłębnej analizie danych. Zasadnicza część rozprawy została poświęcona kryterium separowalności (które jest skuteczną miarą jakości oddzielania obiektów należących do różnych klas), oraz jego zastosowaniom. Podstawowym zastosowaniem jest generowanie[1] drzew decyzji, które mogą być przedstawiane w postaci reguł logiki klasycznej pierwszego rzędu. Innym interesującym zastosowaniem jest przetwarzanie danych celem użycia ich w różnych systemach klasyfikacji, które nie są gotowe do analizy danych w takiej postaci, w jakiej są one przechowywane w bazie: można przy jego pomocy dokonywać dyskretyzacji atrybutów opisujących dane (konwersji z wartości rzeczywistych do symbolicznych) a także uciąglania atrybutów (czyli konwersji w przeciwną stronę). Algorytm tworzenia drzew decyzyjnych pozwala również generować zestawy alternatywnych drzew (zamiast pojedynczego drzewa), czyli dostarczać kilka różnych opisów danych tak, by ekspert mógł ostatecznie zdecydować, który z zestawów reguł najlepiej opisuje badany problem i z porad którego z nich skorzystać. Przedstawiane kryterium separowalności pozwala też budować heterogeniczne drzewa decyzji, które wykorzystują nie tylko źródłowe cechy obiektów, ale np. odległości od pewnych punktów w przestrzeni klasyfikowanych obiektów. Wszystkie z tych metod służą lepszemu poznaniu regularności, które kryją w sobie analizowane dane. Celem nie jest więc system, który byłby uniwersalnym aproksymatorem, lecz narzędzie pozwalające efektywnie i dokładnie opisywać zadania klasyfikacji. A zatem teoretyczna analiza kryterium separowalności ograniczona została do niezbędnego minimum – prezentowane twierdzenia potwierdzają jego podstawowe własności, które choć są dość intuicyjne, to jednak wymagają formalnego uzasadnienia. Skuteczność kryterium została potwierdzona

[1] W dalszej części pracy, tworzenie opisów danych w postaci drzew decyzji (lub reguł klasyfikacji) jest określane równoważnymi terminami *generowanie* oraz *indukcja* drzew (reguł).

empirycznie dla wielu problemów klasyfikacji. Zaprezentowane wyniki potwierdzają, że w praktyce systemy heurystyczne, które nie dają gwarancji optymalności mogą być skuteczniejsze niż metody dysponujące teoretycznymi dowodami uniwersalności. Jest to konsekwencja faktu, że spełnienie warunków niezbędnych do uzyskania optymalnego rozwiązania jest zazwyczaj niemożliwe, co nie pozwala wykorzystać teoretycznie udowodnionych własności.

Samo kryterium separowalności jak i opracowane w oparciu o nie algorytmy są wynikami badań prowadzonych przez autora.

Rozprawa przedstawia także pewne metody indukcji reguł logicznych, wykorzystujące sieci neuronowe. Powszechnie znana sieć MLP (ang. *Multi-Layer Perceptron*) została zmodyfikowana tak, by po procesie adaptacji parametrów mogła być interpretowana logicznie. Algorytm ten powstał w wyniku wspólnej pracy autora oraz Włodzisława Ducha i Rafała Adamczaka. Inny algorytm wykorzystuje metody szukania, by stworzyć sieć MLP o odpowiedniej strukturze i parametrach tak, by w efekcie uzyskać reguły logiczne opisujące problem klasyfikacji.

Praca podejmuje również kilka innych problemów, które pojawiają się w procesach odkrywania wiedzy z danych. Wszystkie zalety opracowanych algorytmów zilustrowane są wynikami jakie metody te dają w zastosowaniu nie tylko do sztucznie postawionych zadań testowych, ale także do realnych problemów.

Część z przedstawianych metod została w ostatnich latach opublikowana w czasopismach bądź zaprezentowana na konferencjach naukowych i opublikowana w materiałach konferencyjnych (lista publikacji i opracowań autora załączona jest w dodatku na str. 168).

Struktura rozprawy jest następująca: Rozdział 2 przedstawia różne aspekty problemów klasyfikacji i indukcji reguł logicznych, oraz prezentuje najpopularniejsze i najczęściej stosowane metody znane z literatury, które wykonują podobne zadania do przedstawianych w pracy nowych rozwiązań, a których wyniki są przedstawiane w porównaniach. Rozdział 3 prezentuje kryterium separowalności i jego własności, a także przedstawia jego zastosowania wraz z licznymi wynikami, które porównuje z wynikami innych systemów. Rozdział 4 opisuje metody wykorzystujące sieci neuronowe do indukcji reguł z danych, a rozdział 5 pewne dodatkowe algorytmy uzupełniające podstawowe zadania.

Rozdział 2

Metody klasyfikacji

Pojęcie *analiza danych* obejmuje bardzo szeroką gamę zadań. Jednym z nich jest zadanie klasyfikacji obiektów, które najczęściej polega na tym, że ekspert na podstawie dostępnych mu danych podejmuje decyzję do jakiej grupy należy zaliczyć dany obiekt. Tak postawione zadanie może być bardzo skutecznie rozwiązywane przez dostępne dzisiaj coraz powszechniej metody inteligencji obliczeniowej. Rolę eksperta pełnią wówczas pewne sparametryzowane modele, które w procesie adaptacji parametrów na podstawie dostępnych danych, stają się nośnikiem wiedzy, która pozwala wykonywać postawione zadanie klasyfikacji.

Skuteczne modele klasyfikujące to takie, które potrafią udzielać poprawnych odpowiedzi także dla danych, które nie były dostępne w czasie uczenia, a pochodzą z tej samej dziedziny. Taka własność nazywana jest *zdolnością generalizacji* czyli umiejętnością uogólniania treści zawartych w analizowanych danych.

2.1 Problem klasyfikacji

Dla ścisłości dalszych rozważań przyjmijmy kilka podstawowych definicji.

Definicja 2.1 ***Przestrzenią klasyfikacji*** *(lub* ***przestrzenią obiektów klasyfikacji****) nazywamy iloczyn kartezjański* $X = X_1 \times X_2 \times \cdots \times X_n$ *skończonej liczby zbiorów, z których każdy jest albo zbiorem liczb rzeczywistych albo pewnym skończonym zbiorem obiektów. Zbiory* X_i *nazywamy* ***cechami*** *lub* ***atrybutami***[1] *tej przestrzeni. Cechy będące zbiorem liczb rzeczywistych nazywamy* ***cechami ciągłymi****, natomiast cechy będące skończonymi zbiorami* ***cechami dyskretnymi*** *lub* ***symbolicznymi****.*

[1] Słowa cecha i atrybut (ang. feature i attribute) są przez niektórych badaczy systemów klasyfikacji rozróżniane, jednak w związku z tym, że intuicje stojące za takim rozróżnianiem nie są oczywiste są one tutaj utożsamiane. Dla klarowności opisów współrzędna wektora należącego do przestrzeni klasyfikacji nazywana będzie wartością danej cechy lub wartością atrybutu.

Uwaga : Każdą przestrzeń klasyfikacji można łatwo przekształcić w przestrzeń cech ciągłych: wystarczy każdą z cech dyskretnych zastąpić cechą ciągłą i reprezentować elementy tej przestrzeni poprzez wzajemnie jednoznaczne odwzorowania cech dyskretnych w podzbiory zbioru liczb rzeczywistych. Technika ta znajduje zastosowanie w sytuacji, kiedy chcemy stosować systemy inteligencji obliczeniowej, których adaptacja wymaga ciągłych wartości wejściowych danych (p. rozdział 3.7). Oczywiście różnych wzajemnie jednoznacznych odwzorowań cechy dyskretnej w podzbiór liczb rzeczywistych jest nieskończenie wiele, więc dla różnych odwzorowań otrzymamy różne reprezentacje tego samego wektora oryginalnej przestrzeni.

Definicja 2.2 ***Problemem (lub zagadnieniem) klasyfikacji*** *nazywamy dowolny skończony podzbiór zbioru $X \times C$, gdzie X jest przestrzenią klasyfikacji, a C jest pewnym skończonym zbiorem obiektów zwanym* ***zbiorem etykiet klas*** *(lub krócej* ***zbiorem klas****) tego problemu klasyfikacji. Dla każdego obiektu $o = (x,c) \in X \times C$, wartość c nazywamy klasą obiektu o i oznaczamy $C(o)$.*

Uwaga : Podobnie jak w przypadku przestrzeni klasyfikacji tak i w przypadku zbioru klas, niektóre systemy wymagają liczbowego przedstawienia poszczególnych klas. Zwykle w takich sytuacjach klasy są kodowane kolejnymi liczbami naturalnymi.

Przykład 2.1 *Niech $X = R^2$ i $C = \{f,t\}$. Zbiór*

$$\{((0,0),f),\ ((0,1),t),\ ((1,0),t),\ ((1,1),f)\}$$

jest problemem klasyfikacji znanym powszechnie pod nazwą problemu XOR (jako, że odpowiada on logicznej funkcji ***albo*** *– ang.* ***exclusive or*** *– w skrócie XOR).*

Definicja 2.3 ***Klasyfikatorem dyskretnym*** *(krótko* ***klasyfikatorem****) nazywamy każdą funkcję $f : X \to C \cup \{C_0\}$, gdzie X jest przestrzenią klasyfikacji, C zbiorem klas oraz $C_0 \notin C$.*

Uwaga : Wartość C_0 wprowadzona zostaje po to, by klasyfikator mógł przyjmować wartości interpretowane jako odpowiedź „nie wiem”.

Definicja 2.4 ***Klasyfikatorem probabilistycznym*** *nazywamy każdą funkcję $f : X \to [0,1]^k$, gdzie X jest przestrzenią klasyfikacji, a zbiór klas C jest k-elementowy.*

Uwagi : Współrzędne wektora $f(x)$ można interpretować jako prawdopodobieństwa przynależności do poszczególnych klas.

Każdemu klasyfikatorowi probabilistycznemu w naturalny sposób odpowiada klasyfikator dyskretny, który jako wartość przyjmuje najbardziej prawdopodobną klasę. W sytuacjach, kiedy więcej niż jednej klasie odpowiada maksymalna wartość, nie opłaca się (z punktu widzenia oceny klasyfikacji) udzielać odpowiedzi „nie wiem" – lepiej wybierać losowo docelową klasę (spośród maksymalnie prawdopodobnych).

Każdemu klasyfikatorowi dyskretnemu w naturalny sposób odpowiada klasyfikator probabilistyczny, który jako wynik daje stosowny wektor binarny.

Definicja 2.5 ***Systemy klasyfikacji*** *to algorytmy, które dla danego problemu klasyfikacji znajdują stosowny klasyfikator.*

Oznaczenia : Przebieg systemu klasyfikacji dla danego problemu nazywa się *procesem uczenia*. Problem klasyfikacji zadany systemowi klasyfikacji nazywa się najczęściej *zbiorem treningowym*. Problemy klasyfikacji wykorzystywane dla sprawdzenia działania klasyfikatora znalezionego przez system nazywa się *zbiorami testowymi*. Niektóre systemy klasyfikacji stosują techniki polegające na wyodrębnianiu ze zbioru treningowego podzbioru wykorzystywanego w procesie szukania optymalnego klasyfikatora oraz podzbioru używanego do wewnętrznego sprawdzania własności znalezionego modelu. Wówczas ten ostatni podzbiór nazywa się *zbiorem walidacyjnym*.

Przykład 2.2 *Jednym z najprostszych systemów klasyfikacji jest algorytm, który znajduje klasyfikator, który jest funkcją stałą, przyjmującą zawsze wartość klasy dominującej w zbiorze treningowym (bądź jedną z dominujących, jeśli więcej niż jedna klasa ma maksymalną liczbę reprezentantów w zbiorze treningowym). Bardziej formalnie, jest to system, który dla danego problemu klasyfikacji* $T = \{(x_i, c_i) : i = 1, \ldots, n\} \subseteq X \times C$ *gdzie* X *jest przestrzenią klasyfikacji,* $n \in N, C = \{C_1, \ldots, C_k\}, k \in N$*, daje w wyniku klasyfikator* $f : X \to C$ *taki, że dla dowolnego* $x \in X$

$$f(x) = \underset{i \in N}{\operatorname{argmax}} |\{(z, c) \in T : c = C_i\}| \tag{2.1}$$

Uwaga : System klasyfikacji przedstawiony w przykładzie 2.2 jest systemem, który definiuje dolną granicę dokładności klasyfikacji sensownych klasyfikatorów. Nazywamy go *klasyfikatorem większościowym* bądź *podstawowym*, a jego dokładność – *dokładnością podstawową* bądź *wartością bazową* (ang. *baseline rate*). Każdy klasyfikator, którego dokładność klasyfikacji jest niższa niż podstawowa powinien być traktowany jako nie kryjący w sobie żadnej wiedzy na temat

zadanego problemu klasyfikacji.

W praktyce zwykle chcemy, aby systemy klasyfikacji dawały w wyniku klasyfikator, który jest w jak największym stopniu zgodny z rozkładem danych, z którego został wygenerowany analizowany problem klasyfikacji. Nie chodzi więc o klasyfikację maksymalnie zgodną z oryginalnym problemem, ale o to by znaleźć klasyfikator, który będzie poprawnie klasyfikował dane z każdego innego problemu klasyfikacji, za którym stoi ten sam rozkład. Zatem chodzi o to, by systemy klasyfikacji posiadały zdolność *generalizacji* problemu. Dlatego też, stosowane algorytmy wykorzystują pewne techniki mające na celu zabezpieczenie się przed nadmiernym dopasowaniem do problemu, którego system się uczy (p. rozdział 2.2.2).

Najskuteczniejszy byłby taki system klasyfikacji, który zdołałby wybrać z całej przestrzeni klasyfikatorów taki, który najlepiej spełniałby pewne założenia. Jednak w praktyce przeszukanie całej przestrzeni ani analityczne wyznaczenie optymalnego klasyfikatora nie są możliwe. Trzeba więc w pewien sposób zredukować złożoność problemu – np. ograniczając przeszukiwaną przestrzeń i szukając optymalnego klasyfikatora w pewnej jej podprzestrzeni. Innym sposobem jest skonstruowanie całkowicie innej przestrzeni, której przeszukiwanie jest łatwiejsze, a istnieją podstawy by twierdzić (albo wręcz dowody na to), że optymalne punkty w tej przestrzeni odpowiadają optymalnym rozwiązaniom oryginalnego zadania. Sztucznie skonstruowana przestrzeń może stwarzać dużo większe szanse znalezienia atrakcyjnego rozwiązania nawet jeśli te same klasyfikatory mogą mieć w niej wiele różnych reprezentacji, co w pierwszej ocenie wydaje się utrudniać a nie ułatwiać zadanie. Systemy klasyfikacji najczęściej definiują przestrzeń możliwych rozwiązań poprzez założenie pewnej sparametryzowanej postaci klasyfikatora. W procesie adaptacji dostrajają parametry modelu tak, by uzyskać jak najlepszy (w rozumieniu założonej funkcji celu) klasyfikator. Adaptacja parametrów może przyjmować bardzo różne postaci, z których najczęściej stosowane opisane są w podrozdziale 2.4.

2.2 Wyniki klasyfikacji

Celem systemów klasyfikacji jest dokonywanie jak najlepszej klasyfikacji danych. Słowo „najlepszej” nie jest jednak w tym kontekście jednoznaczne. Różne są wymagania co do wyników klasyfikacji – czasami interesuje nas po prostu jak najwyższa dokładność (tj. liczba poprawnie klasyfikowanych obiektów), ale często istotne jest dla jakich danych popełniane są błędy, bo np. ważniejszym jest nie pomylić się, kiedy pacjent rzeczywiście jest chory (i wykryć u niego chorobę), niż kiedy jest zdrowy (wówczas pomyłka oznacza w najgorszym wypadku niepo-

trzebny koszt dodatkowych badań). Tak czy inaczej, zawsze jest bardzo istotne, aby model, który znajdziemy, uogólniał analizowany problem i nadawał się do klasyfikacji danych, które nie były mu dostępne w trakcie uczenia. Dlatego też każdy system klasyfikacji powinien być wyposażony w mechanizmy zapewniające, że w trakcie uczenia nie „przeuczy" się (tzn. nie dopasuje nadmiernie do danych treningowych), ale wyciągnie z nich rzeczywiście istotne informacje, które dadzą mu zdolność generalizacji.

2.2.1 Wartości oceniające działanie klasyfikatora

Podstawowym kryterium oceny klasyfikatorów jest dokładność klasyfikacji (poprawność). Definiuje się jednak wiele innych wielkości, które są niezwykle cenne z punktu widzenia eksperta próbującego zrozumieć sposób działania klasyfikatora i ocenić jego przydatność w konkretnych zastosowaniach.

Definicja 2.6 *Niech X będzie przestrzenią klasyfikacji, $C = \{C_1, \ldots, C_k\}$ zbiorem klas, $C_0 \notin C$, $T = \{(t_i, c_i) : i = 1, \ldots, n\} \subseteq X \times C$ problemem klasyfikacji ($k, n \in N$), oraz $f : X \to C \cup \{C_0\}$ klasyfikatorem.*
Dokładnością (poprawnością) klasyfikacji *(ang. accuracy) klasyfikatora f dla zbioru T nazywamy*

$$\mathsf{Acc}(f,T) \stackrel{def}{=} P(f(t) = c | (t,c) \in T).$$

Błędem klasyfikacji *klasyfikatora f dla zbioru T nazywamy*

$$\mathsf{Err}(f,T) \stackrel{def}{=} P(f(t) \neq c | (t,c) \in T).$$

Dla danej klasy C_i definiujemy ponadto następujące wartości:

$$\begin{aligned}
\mathsf{Pos}(C_i,T) &\stackrel{def}{=} |\{(t,c) \in T : c = C_i\}| \\
\mathsf{Neg}(C_i,T) &\stackrel{def}{=} |\{(t,c) \in T : c \neq C_i\}| \\
\mathsf{True}(f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) = c\}| \\
\mathsf{False}(f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) \neq c\}| \\
\mathsf{TP}(C_i,f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) = c \text{ oraz } c = C_i\}| \\
\mathsf{FP}(C_i,f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) = c \text{ oraz } c \neq C_i\}| \\
\mathsf{TN}(C_i,f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) \neq c \text{ oraz } c \neq C_i\}| \\
\mathsf{FN}(C_i,f,T) &\stackrel{def}{=} |\{(t,c) \in T : f(t) \neq c \text{ oraz } c = C_i\}|
\end{aligned}$$

Wrażliwością (ang. sensitivity) *klasyfikatora f na klasę C_i dla zbioru T nazywamy*

$$\mathsf{Se}(C_i, f, T) \stackrel{def}{=} P(f(t) = C_i | C(t) = C_i) = \frac{\mathsf{TP}(C_i, f, T)}{\mathsf{Pos}(C_i, T)}$$

Znamiennością (ang. specificity) *klasyfikatora f dla klasy C_i i zbioru T nazywamy*

$$\mathsf{Sp}(C_i, f, T) \stackrel{def}{=} P(f(t) \neq C_i | C(t) \neq C_i) = \frac{\mathsf{TN}(C_i, f, T)}{\mathsf{Neg}(C_i, T)}$$

Wniosek 2.1 *Wprost z definicji wynikają następujące równości:*

$$\mathsf{Acc}(f, T) = 1 - \mathsf{Err}(f, T)$$

$$\mathsf{Acc}(f, T) = \frac{\mathsf{True}(f, T)}{|T|}$$

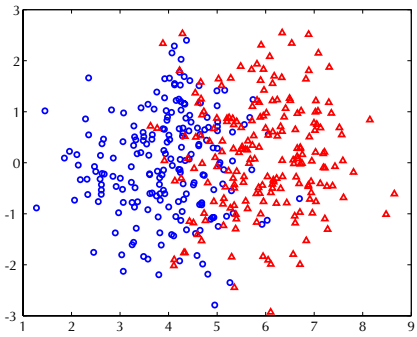
$$\mathsf{Err}(f, T) = \frac{\mathsf{False}(f, T)}{|T|}$$

$$\mathsf{Pos}(C_i, T) = \mathsf{TP}(C_i, f, T) + \mathsf{FN}(C_i, f, T)$$

$$\mathsf{Neg}(C_i, T) = \mathsf{TN}(C_i, f, T) + \mathsf{FP}(C_i, f, T)$$

Uwagi : Wrażliwość określa na ile klasyfikator jest zdolny do wykrywania przypadków z danej klasy (jest to prawdopodobieństwo warunkowe klasyfikacji do wybranej klasy pod warunkiem, że obiekt rzeczywiście do niej należy).
Znamienność określa na ile decyzja klasyfikatora o przynależności do wybranej klasy jest charakterystyczna wyłącznie dla tej klasy (jest to uzupełnienie prawdopodobieństwa warunkowego klasyfikacji do wybranej klasy pod warunkiem, że obiekt do tej klasy nie należy).
Bez utraty ogólności można ograniczyć analizę wrażliwości i znamienności do problemów dwuklasowych.

Krzywe ROC. W niektórych zastosowaniach (szczególnie w medycynie) bardzo często dla wizualnej oceny działania klasyfikatora (a dokładniej rodziny klasyfikatorów) kreśli się tzw. *krzywe ROC* (ang. *Receiver Operation Characteristic curve*) [59, 127]. Wykreślenie takiej krzywej polega na umieszczeniu na wykresie dla każdego klasyfikatora punktu o odciętej równej wrażliwości klasyfikatora dla danej klasy oraz rzędnej równej uzupełnieniu znamienności do 1 (bądź znamienności z odwróconym porządkiem) i przeprowadzeniu krzywej przez naniesione punkty.

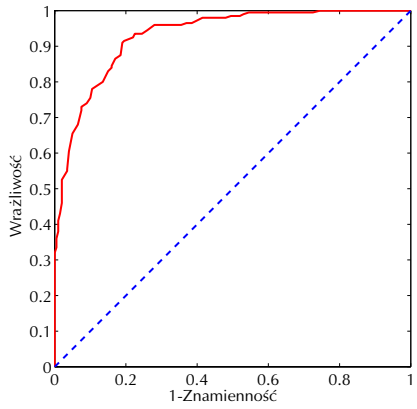


Rysunek 2.1: Dwie klasy o nakładających się rozkładach normalnych

Przykład 2.3 *Niech* $X = R^2$ *oraz* $C = \{1,2\}$. *Niech* $T_1 \in X$ *będzie zbiorem* 200 *punktów wygenerowanych z rozkładem* $N((4,0),1)$ *(normalnym o wartości oczekiwanej* $(4,0)$ *i odchyleniu standardowym* 1*) oraz* $T_2 \in X$ *zbiorem* 200 *punktów z rozkładu* $N((6,0),1)$ *oraz* $T = \{(t,1) : t \in T_1\} \cup \{(t,2) : t \in T_2\}$. *Tak skonstruowany zbiór* T *przedstawiony został na rysunku 2.1 (punktom z klasy* 1 *odpowiadają niebieskie kółka, a tym z klasy* 2 *czerwone trójkąty).*
Dla dowolnej wartości rzeczywistej S można stworzyć klasyfikator

$$f_S(x) = \begin{cases} 1 & \text{jeżeli } x_1 \leqslant S \\ 2 & \text{w przeciwnym przypadku} \end{cases} \tag{2.2}$$

Krzywą ROC dla takiej rodziny klasyfikatorów przedstawia rysunek 2.2. Punkt



Rysunek 2.2: Krzywa ROC dla klasyfikatorów 2.2

$(0,1)$ *reprezentuje klasyfikator dla* $S = 0$. *Taki klasyfikator przyporządkowuje każ-*

demu z wektorów zbioru T klasę 2. *Klasyfikator stały o wartości klasy* 1 *dostajemy np. dla* $S = 10$ *i jest on reprezentowany przez punkt* $(1,0)$.

Oprócz zilustrowanych w powyższym przykładzie punktów $(0,1)$ i $(1,0)$ na szczególną uwagę zasługują także punkty leżące w blisko przekątnej (oznaczonej na rysunku 2.2 niebieską, przerywaną linią), oraz punkt $(1,1)$. Okolice przekątnej reprezentują klasyfikatory, których jakość działania jest zbliżona do klasyfikatorów działających losowo. Poniżej tej przekątnej reprezentowane są klasyfikatory mniej dokładne niż losowe, a więc nie zasługujące na jakiekolwiek zastosowania. Najwyższa (stuprocentowa) poprawność umieszcza klasyfikator w punkcie $(0,1)$. Najlepszym klasyfikatorom będą więc odpowiadały okolice tego właśnie punktu.

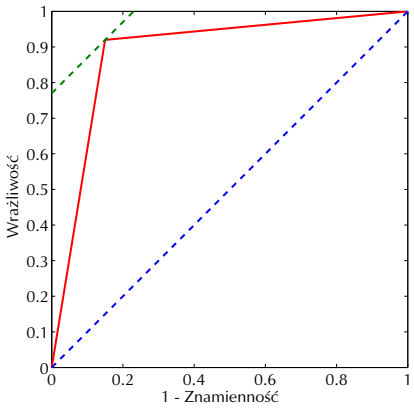
W interpretacji krzywych ROC zwraca się uwagę przede wszystkim na pole powierzchni obszaru pod krzywą (ang. *Area Under Curve* – AUC). Im większe pole, tym „lepsza" rodzina klasyfikatorów. W praktyce najczęściej jako rodzinę klasyfikatorów wybiera się wszystkie klasyfikatory uzyskane z danego klasyfikatora probabilistycznego przez zadanie pewnego progu decyzji tzn. dla klasyfikatora f, zbioru klas $C = \{\text{prawda}, \text{fałsz}\}$ oraz progu $S \in [0,1] \cup \{1+\varepsilon\}$ (dla pewnego $\varepsilon > 0$) mamy klasyfikator dyskretny:

$$g(x) = \begin{cases} \text{prawda} & \text{jeżeli } f(x)_i \geqslant S \\ \text{fałsz} & \text{w przeciwnym przypadku} \end{cases} \tag{2.3}$$

W ten sposób można dany klasyfikator probabilistyczny poddać analizie wiarygodności na podstawie krzywej ROC. Często porównuje się w ten sposób różne klasyfikatory by wyznaczyć najbardziej interesujący, czyli ten, który daje największe pole pod krzywą.

Istotnie maksymalizacja AUC jest mocno skorelowana z dokładnością klasyfikacji, a równocześnie z bezbłędnością tych odpowiedzi klasyfikatora, które są bliskie zeru bądź jedynce. Jeśli mamy do czynienia z klasyfikatorem dyskretnym, to rodzina klasyfikatorów, którą stworzymy dla odpowiadającego mu klasyfikatora probabilistycznego da krzywą ROC złożoną z dwóch odcinków jak na rysunku 2.3. Każdy klasyfikator, dla którego punkt (1-znamienność,wrażliwość) leży na odcinku zaznaczonym zieloną przerywaną linią ma dokładnie takie samo pole pod krzywą jak przedstawiony klasyfikator. Innymi słowy dla klasyfikatora dyskretnego maksymalizacja AUC to maksymalizacja sumy znamienności i wrażliwości, a więc jest to prawie to samo, co po prostu maksymalizacja poprawności klasyfikacji (dla zbiorów z dwiema równolicznymi klasami jest to dokładnie to samo).

Dla klasyfikatorów probabilistycznych, które w wektorze wynikowym dają wartości różne od zera i jedynki związki pomiędzy polem pod krzywą ROC a poprawnością klasyfikacji nie są tak proste. Niech T będzie zadanym problemem klasyfikacji. Dla m klasyfikatorów $f_1, \ldots, f_m$ takich, że $0 = \mathsf{Se}(\text{prawda}, f_1, T) \leqslant$



Rysunek 2.3: Krzywa ROC dla klasyfikatora dyskretnego

$\cdots \leqslant \mathsf{Se}(\text{prawda}, f_m, T) = 1$ oraz $1 = \mathsf{Sp}(\text{prawda}, f_1, T) \geqslant \cdots \geqslant \mathsf{Sp}(\text{prawda}, f_m, T) = 0$ mamy

$$\mathsf{AUC} = -\frac{1}{2}\sum_{j=2}^{m}(\mathsf{Sp}_j\mathsf{Se}_{j-1} - \mathsf{Sp}_{j-1}\mathsf{Se}_j), \tag{2.4}$$

gdzie

$$\mathsf{Sp}_j = \mathsf{Sp}(\text{prawda}, f_j, T) \quad \text{oraz} \quad \mathsf{Se}_j = \mathsf{Se}(\text{prawda}, f_j, T).$$

Analizę AUC stosuje się do poszukiwania optymalnego progu dla rozróżnienia dwóch klas [38], dla porównywania klasyfikatorów [83, 84, 34], a także dla tworzenia systemów hybrydowych zdolnych dopasowywać klasyfikację do preferencji użytkownika w sytuacjach, kiedy trudno a priori ocenić koszty odpowiednich błędów klasyfikacji [60, 161]. Poza nielicznymi wyjątkami [149] trudno doszukać się w literaturze systemów, które używałyby takiego kryterium bezpośrednio w procesach optymalizacyjnych szukających klasyfikatorów.

Czytelność rozwiązań. W niektórych zastosowaniach jednym z podstawowych wymogów stawianych systemom inteligencji obliczeniowej podejmującym problemy klasyfikacji obiektów jest czytelność funkcji decyzyjnych. W ocenie przydatności klasyfikatora bierze się wówczas pod uwagę możliwość interpretacji działania klasyfikatora, która może być bardziej istotna niż osiąganie wysokiej poprawności klasyfikacji. Wiele systemów klasyfikacji (zwłaszcza sieci neuronowe) buduje modele, które są opisywalne funkcjami matematycznymi, których poziom złożoności przekracza możliwości interpretacyjne człowieka. Przybierają więc one postać *czarnych skrzynek*, które po uzyskaniu na wejściu wektora danych dają na wyjściu wartość klasy bez żadnych wyjaśnień dlaczego taka a nie inna decyzja została w danym przypadku podjęta.

Szczególnie w zastosowaniach medycznych, ale także ekonomicznych czy marketingowych, cenne jest uzyskanie od systemu klasyfikacji wyjaśnienia podjętej decyzji.

Lekarz, który chce skorzystać z pomocy metod inteligencji obliczeniowej przy wystawianiu diagnozy, chciałby mieć możliwość weryfikacji poprawności uzyskiwanych podpowiedzi. Jeśli wyjaśnienie przedstawione jest w czytelnej dla niego formie, może z łatwością porównać sposób wnioskowania systemu z własną wiedzą i doświadczeniem, a więc będzie chętnie z takiej pomocy korzystał. Żaden lekarz nie chciałby brać na siebie odpowiedzialności za diagnozę wystawioną przez czarną skrzynkę, więc zastosowania medyczne powinny skupiać się na wspomaganiu decyzji lekarzy przez łatwo interpretowalne podpowiedzi.

W bankowości, gdzie celem jest np. rozpoznawanie podejrzanych transakcji, czy w marketingu, gdzie wartościowe jest rozpoznanie pewnych profili klientów tak, by lepiej trafiać ze składanymi ofertami i redukować w ten sposób koszty, również bardzo istotne jest, by odkrywana z baz danych wiedza przybierała formę czytelną dla eksperta z danej dziedziny.

Najłatwiej interpretowalną przez człowieka formą wiedzy wydają się być reguły wnioskowania zdefiniowane w logice klasycznej. Systemy, których funkcja decyzyjna może przybierać postać takich reguł mają więc tutaj do spełnienia olbrzymią rolę. Istotne jest oczywiście nie tylko to czy daje się daną wiedzę zapisać w postaci reguł, ale również na ile te reguły są czytelne. Bardzo liczne zestawy reguł klasyfikacji nie spełnią zadania wyjaśniania podejmowanych decyzji.

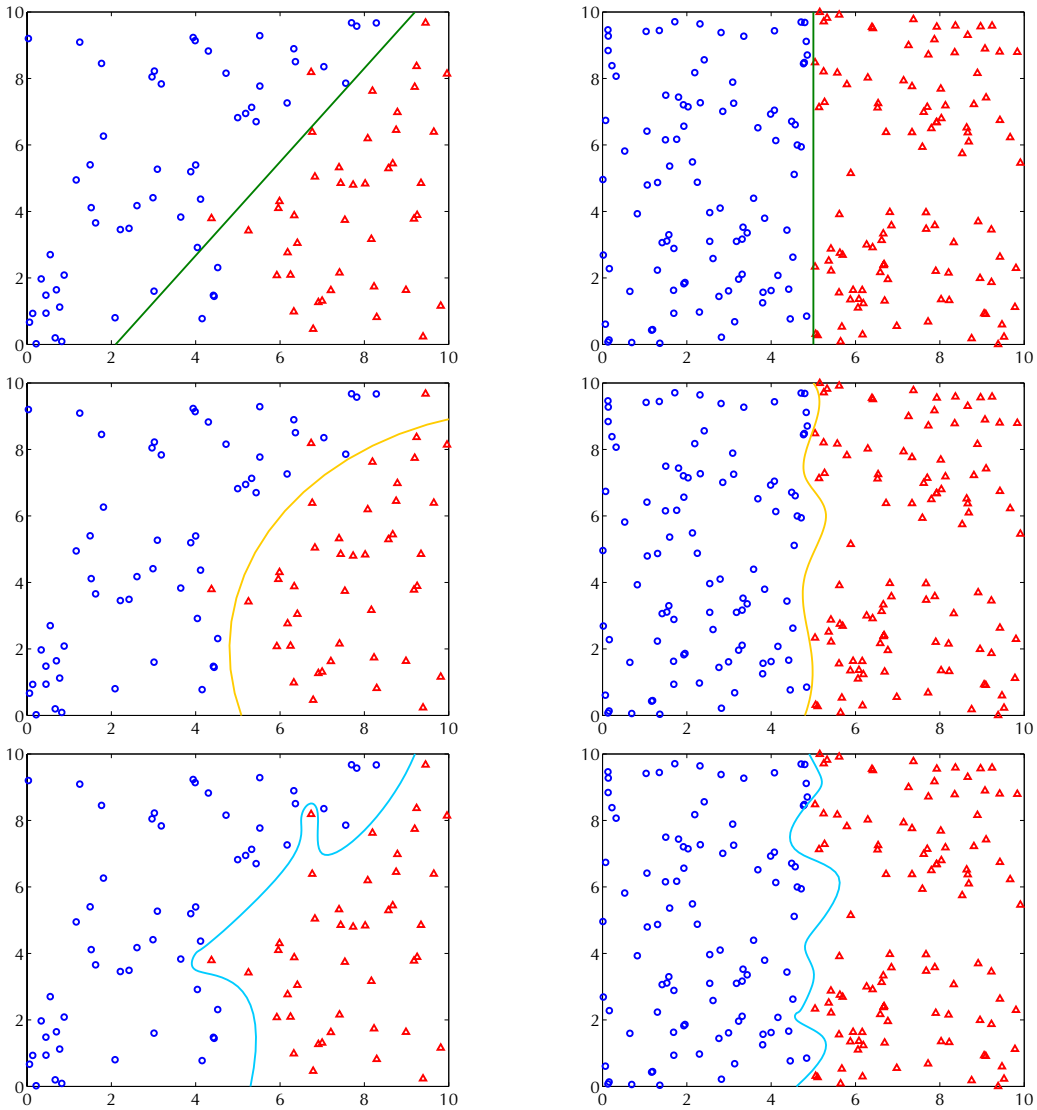
Alternatywą dla reguł logiki klasycznej są *reguły rozmyte*. Systemy oparte na logice rozmytej produkują najczęściej dość duże zestawy reguł. Ich interpretacja, mimo to, nie jest trudna, kiedy rozpatrujemy klasyfikację pojedynczego wektora – wówczas możemy się skupić tylko na tych regułach, które dla jego klasyfikacji mają istotne znaczenie. W pewnych zastosowaniach reguły rozmyte mogą być bardziej naturalne niż reguły logiki dwuwartościowej, w innych preferowanym rozwiązaniem są reguły w klasycznej postaci. Tak czy inaczej, zestawy nielicznych reguł klasyfikacji pozwalają dodatkowo na globalną interpretację sposobu rozróżniania obiektów przez system.

2.2.2 Generalizacja

Niezależnie od szczegółów sposobu oceny jakości klasyfikatorów nadrzędnym celem jest zdolność generalizacji, czyli umiejętność formułowania ogólnych twierdzeń na podstawie analizowanych przykładów. Jednym ze sposobów uczenia może być uczenie się „na pamięć” – wystarczy zapamiętać całość zbioru treningowego, by klasyfikować go z maksymalną dokładnością, jednak klasyfikacja przypadków nie zawartych w zbiorze treningowym (nie oglądanych w trakcie uczenia)

będzie wówczas niemożliwa. Jeśli model nadmiernie dopasuje się do danych treningowych, to poprawność dla nowych danych będzie bardzo niska.

Rozważmy zadania klasyfikacji danych w R^2 przedstawione na rysunku 2.4. Trzy wykresy umieszczone w kolumnie przedstawiają trzy różne granice decyzji



Rysunek 2.4: Dwa zadania klasyfikacji w R^2

klasyfikatorów dla tego samego zadania klasyfikacji.

W przypadku zadania przedstawionego wykresami umieszczonymi z lewej strony, klasyfikator z liniową granicą decyzji popełnia na tym zbiorze 7 błędów,

z owalną granicą – 3, a z granicą przedstawioną na najniższym rysunku rozdziela klasy bezbłędnie. Fakt ten nie upoważnia nas jednak do twierdzenia, że ten trzeci klasyfikator jest obiektywnie najlepszy – wśród nowych punktów wygenerowanych z tego samego rozkładu odsetek popełnianych błędów będzie prawdopodobnie wyższy niż w przypadku klasyfikatora o owalnej granicy decyzji.

Drugie zadanie (przedstawione wykresami z prawej strony) jest dla zbioru treningowego rozwiązywane bezbłędnie przez wszystkie trzy przedstawione klasyfikatory. Górny wykres przedstawia bardzo prostą granicę decyzyjną, dolny – granicę klasyfikatora, który dla zbioru treningowego daje możliwie duże marginesy.

Już od czasów średniowiecza znana jest maksyma zwana *brzytwą Ockhama*, według której „nie należy mnożyć bytów ponad konieczność". Znajduje ona zastosowanie i tutaj przekładając się na wybór tak prostych rozwiązań jak to tylko możliwe. Zbyt złożone modele mają tendencję do nadmiernego dopasowywania się do danych treningowych, co odpowiada uczeniu się „na pamięć" i jest często określane jako *przeuczenie się* systemu. Jest to sytuacja analogiczna do rozwiązywania problemów aproksymacji: np. aproksymując wielomianem można bardzo dobrze dopasować się do dostępnych danych kiedy użyjemy wielomianu wysokiego stopnia, jednak taki wielomian może bardzo nietrafnie przybliżać funkcję w innych punktach niż reprezentowane próbką danych.

Nie istnieje jednoznaczna miara, która pozwoliłaby dla danego zbioru treningowego oraz modelu klasyfikacyjnego efektywnie twierdzić, czy model jest optymalny z punktu widzenia generalizacji.

Zakładając jako podstawę analizy pewien zbiór hipotez H, można rozważać teoretycznie, która z hipotez daje maksymalne prawdopodobieństwo słuszności dla danego zbioru treningowego D. Prowadzi to do wyznaczenia hipotezy maksymalizującej prawdopodobieństwo *a posteriori*:

$$h_{MAP} = \operatorname*{argmax}_{h \in H} P(h|D) = \operatorname*{argmax}_{h \in H} P(D|h)P(h) \tag{2.5}$$

Można również użyć równoważnego zapisu:

$$h_{MAP} = \operatorname*{argmax}_{h \in H} \left[\log_2 P(D|h) + \log_2 P(h)\right] = \operatorname*{argmin}_{h \in H} \left[-\log_2 P(D|h) - \log_2 P(h)\right]. \tag{2.6}$$

A zatem korzystając z języka teorii informacji można powiedzieć, że optymalną hipotezą jest ta, która minimalizuje sumę długości opisów teorii h (przy optymalnym kodowaniu hipotez w H) oraz danych D przy założeniu hipotezy h (również dla optymalnego kodowania). W ogólnym przypadku oznaczając przez $L_C(X)$ długość opisu obiektu X przy kodowaniu C, dla danych kodowań hipotez C_H oraz danych C_h (przy założeniu hipotezy h) można określić zasadę minimalnej długości

opisu (ang. *Minimum Description Length* – MDL) [155]:

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} \left[L_{C_H}(h) + L_{C_h}(D) \right]. \tag{2.7}$$

Zasada ta jest teoretycznym uzasadnieniem faktu, że krótsze opisy klasyfikacji dają większe prawdopodobieństwo dobrej generalizacji. Dokładniejsze studium tematu można znaleźć np. w [131].

Na bazie tej samej teorii można również zdefiniować Optymalny Klasyfikator Bayesowski (ang. *Bayes Optimal Classifier*), który maksymalizuje prawdopodobieństwo poprawnej klasyfikacji nowych danych przy zadanych: zbiorze treningowym D, przestrzeni hipotez H oraz prawdopodobieństwach *a priori* poszczególnych hipotez:

$$OKB(x) = \underset{c \in C}{\operatorname{argmax}} \sum_{h \in H} P(c|h,x)P(h|D). \tag{2.8}$$

W praktyce konstrukcja takiego klasyfikatora jest niemożliwa między innymi z powodu zbyt dużej liczby hipotez (być może nieskończonej), które należałoby rozważyć w celu klasyfikacji każdego przypadku, czy też z powodu nieznajomości „probabilistycznej struktury" problemu.

Techniki poprawiające generalizację

Z podobnych powodów, dla których w praktyce niemożliwe jest zbudowanie optymalnego klasyfikatora Bayesowskiego, również zastosowanie kryterium MDL do szukania optymalnych modeli często jest nierealne. Podejmowane były różne próby wykorzystania MDL do kontroli złożoności modeli, między innymi także drzew decyzji [154, 125]. Wykonane eksperymenty pokazują, że można w ten sposób osiągnąć podobne wyniki jak stosując podejścia heurystyczne.

W metodach sieci neuronowych korzysta się z różnych form regularyzacji [9], których cel jest zbliżony do celu kryteriów takich jak MDL.

W systemach generujących modele metodami szukania, kontrola złożoności modelu odbywać się może poprzez analizę generalizacji modeli budowanych na pewnych podzbiorach zbioru treningowego i testowanych na wydzielonej z tego zbioru części nie używanej do uczenia. Zastosowanie znajdują tutaj takie metody jak kroswalidacja (p. rozdział 2.2.3) czy algorytmy oparte na technice o angielskiej nazwie *bootstrap* (p. rozdział 2.13).

2.2.3 Techniki oceny klasyfikatorów

W literaturze można znaleźć mnóstwo publikacji na temat algorytmów klasyfikacji. Prezentowane są w nich wyniki klasyfikacji, a czasami także porównania

z innymi metodami. Niestety, bardzo rzadko można spotkać rzetelne porównania wyników uzyskanych różnymi systemami.

Sprawiedliwym porównaniem jakości systemów mogłoby być porównanie wartości oczekiwanych poprawności budowanych przez nie modeli. Na ogół nie jest to możliwe, a jednym z głównych powodów jest niedostępność informacji na temat rzeczywistego rozkładu danych, która uniemożliwia analityczne wyliczenia. Dobre oszacowanie rozkładu wyników można by uzyskać, gdyby możliwe było generowanie dowolnie dużych ilości danych i wykonanie odpowiednio dużej liczby procesów trenowania oraz testowania modeli. Jednak w praktyce, dysponujemy zwykle skończonym zbiorem wektorów, który znacznie ogranicza możliwości testowania.

Wciąż trwa dyskusja na temat sposobów porównywania różnych systemów. Jak dotąd nie wypracowano metod porównawczych, które znalazłyby powszechne uznanie. Każda z metod ma swoich zwolenników i przeciwników, zalety i wady, przykłady trafnych spostrzeżeń i przypadków rozstrzyganych w sposób kontrowersyjny.

Stosunkowo najłatwiejsze dla porównań są dane, w których wydzielono części treningową i testową. Wówczas zwykle jako wynik podaje się pojedynczą liczbę określającą dokładność klasyfikacji na danych testowych, nie widzianych (od ang. *unseen*) przez system w czasie procesu adaptacji parametrów. Ponieważ systemy klasyfikacji rzadko są deterministyczne, pełniejszą informację niosą ze sobą dwie wartości – średnia dokładność klasyfikacji dla zbioru testowego oraz jej wariancja. Wariancja ta nie jest jednak dobrym oszacowaniem rzeczywistej wariancji, bo jest wynikiem niestabilności systemu a ignoruje wpływ różnic w zestawach danych na kształt uzyskiwanych modeli.

Niestety nawet w przypadku wyraźnie wydzielonej części testowej łatwo o nie całkiem rzetelne wyniki. Jeśli dobór parametrów systemu nie jest w pełni zautomatyzowany, bywa tak, że publikuje się wyniki przedstawiając dla jakich wartości parametrów uzyskano prezentowaną jakość klasyfikacji. Może to oznaczać wykorzystanie zbioru testowego na poziomie *meta*, czyli na poziomie szukania odpowiedniego modelu (najczęściej wykonywanego ręcznie przez badacza), a to oczywiście jest poważne nadużycie. Najrzetelniejsze wydają się być takie porównania, w których stosuje się systemy z domyślnymi parametrami tak jak np. w przypadku wyników zebranych w ramach projektu STATLOG [130]. Takie wyniki mogą jednak być dalekie od optymalnych dla danego klasyfikatora. Niestety, bardzo wiele publikacji na temat systemów klasyfikacji nosi znamiona pewnych drobnych lub poważniejszych błędów, więc można mieć poważne wątpliwości co do rzetelności prezentowanych w nich wyników.

Bardzo trafnym podejściem do problemu porównania różnych systemów są zawody organizowane pod nazwą *Unlabeled Data Supervised Learning Competition* w ramach konferencji NIPS (*Neural Information Processing Systems*) [116].

Organizatorzy nie dają możliwości wykorzystania etykiet klas przypisanych wektorom zbiorów testowych, ponieważ etykiety te nie są udostępniane razem ze zbiorami testowymi – pozostają jedynie do dyspozycji organizatorów.

Wyniki pokazują jednak, że i tutaj można znaleźć sposób do nadużyć. Niektórzy z zawodników przesyłali organizatorom do oceny wyniki swoich systemów ponad 100 razy (p. tabela 3.12 na stronie 111). Wielokrotne wysyłanie rezultatów i otrzymywanie informacji o jakości klasyfikacji poszczególnych modeli daje nam również pewne informacje o klasyfikacji zbioru testowego. Łatwo skonstruować algorytm, który poprzez wielokrotne budowanie modeli i analizę uzyskanych rezultatów pozwoli wiernie odtworzyć przypisane poszczególnym wektorom etykiety.

Cross Validation. Kiedy dane dostępne są w jednym zbiorze, bez wyraźnego podziału na zbiory treningowy i testowy, często stosuje się technikę o angielskiej nazwie *cross validation*. Polscy statystycy nazywają ją czasami *testowaniem* lub *sprawdzaniem krzyżowym* albo *walidacją krzyżową*, jednak najczęściej używanym terminem jest kalka z angielskiego – *kroswalidacja*. *n*-krotne testowanie krzyżowe polega na losowym podziale zbioru danych D na n rozłącznych i (w miarę możliwości) równolicznych części $D_1, \ldots, D_n$, oraz uczeniu systemu klasyfikacji n razy, używając w i-tym przebiegu zbioru $D \setminus D_i$ jako zbioru treningowego, a zbioru D_i jako testowego. Głównym wynikiem kroswalidacji jest średnia (dla wszystkich n przebiegów) dokładność klasyfikacji zbioru testowego. Oczywiście wariancja jakości klasyfikacji zbioru testowego jest również bardzo cennym wynikiem charakteryzującym działanie danego systemu.

Dla zwiększenia wiarygodności wyników kroswalidacji wykonuje się wielokrotnie (dla różnych podziałów zbioru D) komplet n przebiegów i podaje średnią wartość średniej dokładności na zbiorze testowym. W przypadku podawania również wariancji wyników testu należy dla uniknięcia nieporozumień wyraźnie wskazywać czy jest to wariancja średniego wyniku testowego kroswalidacji, czy może średnia wariancja wyników testowych (oczywiście ta pierwsza jest zwykle zdecydowanie mniejszą wartością, jednak ta druga jest bardziej wartościowa z punktu widzenia oceny systemów).

Kroswalidacja warstwowa lub *stratyfikowana* (ang. *stratified cross validation*) to rodzaj kroswalidacji, w którym losowy podział jest wykonywany w taki sposób, aby w poszczególnych podzbiorach zachowane były proporcje liczebności klas z oryginalnego zbioru. Jest to więc równoważne dokonaniu całkowicie losowego podziału dla każdej z klas (zbioru wektorów reprezentujących tę klasę) i stosownemu połączeniu otrzymanych zbiorów.

Inna technika z tej samej grupy nosi nazwę *kroswalidacji Monte Carlo* i polega na losowym wybieraniu zadanej parametrem (najczęściej ok. 70%) części danych

do zbioru treningowego, podczas gdy pozostała część służy jako zbiór testowy. Wielokrotne powtarzanie tego testu odbywa się przy niezależnych losowaniach w każdym przebiegu (w przeciwieństwie do klasycznej wersji kroswalidacji). Zaleca się powtarzanie tego testu przynajmniej 30 razy, co pozwala zakładać słuszność przybliżenia wyników rozkładem normalnym.

Leave-one-out (w skrócie L1O), to test walidacji krzyżowej z liczbą przebiegów równą liczbie wektorów danych. W każdym przebiegu trenujemy system na zbiorze powstałym z całego zbioru danych przez usunięcie jednego z wektorów, który z kolei stanowi jednoelementowy zbiór testowy. Choć wydawałoby się, że tak drobne zaburzenie rozkładu jakim jest usunięcie jednego z wektorów nie może mieć istotnego wpływu na ostateczny wynik czyli, że oszacowany przez L1O błąd będzie dobrze określał zdolności generalizacyjne systemu, to jednak można znaleźć przykłady pokazujące błędność tej tezy. Kohavi [110] pokazuje jako przykład zbiór danych *Iris*, który zawiera po 50 wektorów z trzech różnych klas. Dla klasyfikatora większościowego (przykład 2.2) oczekiwalibyśmy oceny poprawności w okolicach 33%, jednak test L10 na skutek zubożenia w zbiorze treningowym o 1 wektor klasy wektora testowego nigdy nie daje poprawnej odpowiedzi dając 0% poprawności z wariancją 0. W przypadku mniejszej wartości liczby przebiegów kroswalidacji wynik jest zdecydowanie lepszy, aczkolwiek również nieco odbiega od oczekiwanej dokładności. Oczywiście zastosowanie kroswalidacji stratyfikowanej daje w tym przypadku wynik doskonale zgodny z oczekiwaniami.

Porównywanie wyników dla różnych systemów – test t Studenta. Najczęściej porównanie różnych systemów ogranicza się do porównania średniej poprawności klasyfikacji – za lepszy uznaje się ten system, który da wyższą wartość. Jednakże porównanie pojedynczych wartości jest mało wiarygodne. Aby rzetelniej porównać wyniki należałoby ocenić prawdopodobieństwo tego, że jeden z systemów zadziała lepiej niż drugi. W naturalny sposób znajdują tutaj zastosowanie metody weryfikacji hipotez statystycznych [122, 18, 131]. Przy założeniu rozkładu normalnego wyników klasyfikacji każdego z systemów oraz niezależność zmiennych odpowiadających tym systemom, wystarczyłoby uwzględnić ich wartości oczekiwane i odchylenia standardowe by wyliczyć stosowne prawdopodobieństwo. W związku z na ogół małą liczebnością próbek szacujących błędy (brakiem możliwości rzetelnej oceny wariancji rozkładu różnic), bardziej zasadne jest użycie testu t Studenta dla zweryfikowania hipotezy o statystycznej istotności (bądź nie) różnic w wynikach badanych dwóch systemów.

Algorytm 2.1 (Test t Studenta istotności różnic wyników klasyfikacji)

- ***Dane:*** *Wyniki* $A_1,\ldots,A_n,B_1,\ldots,B_m \in [0,1]$, $n,m \in N$, *dla systemów klasyfikacji A i B, poziom pewności odpowiedzi p.*

◄ ***Wynik:*** *Odpowiedź na pytanie, czy system A daje z prawdopodobieństwem co najmniej p lepszą jakość klasyfikacji niż system B.*

1. *Szacujemy średnie i wariancje rozkładów wyników dla systemów A i B:*

$$M_A = \frac{1}{n}\sum_{i=1}^{n} A_i \qquad \sigma_A^2 = \frac{1}{n-1}\Big(\sum_{i=1}^{n} A_i^2 - \frac{1}{n}(\sum_{i=1}^{n} A_i)^2\Big)$$

$$M_B = \frac{1}{m}\sum_{i=1}^{m} B_i \qquad \sigma_B^2 = \frac{1}{m-1}\Big(\sum_{i=1}^{m} B_i^2 - \frac{1}{m}(\sum_{i=1}^{m} B_i)^2\Big)$$

2. *Szacujemy wariancję dla całej populacji jako średnią ważoną dla populacji A i B:*

$$\sigma^2 = \frac{(n-1)\sigma_A^2 + (m-1)\sigma_B^2}{(n-1)+(m-1)}$$

3. *Szacujemy wariancję zmiennej $M_A - M_B$:*

$$\sigma_M^2 = \frac{\sigma^2}{n} + \frac{\sigma^2}{m}$$

4. *Wyznaczamy wartość*

$$t = \frac{M_A - M_B}{\sigma_M}$$

5. *Jeśli $t \geqslant t_{p,n+m-2}$, gdzie $t_{p,n+m-2}$ jest kwantylem dla prawdopodobieństwa p rozkładu t Studenta o $n+m-2$ stopniach swobody, to kończymy odpowiadając, że prawdopodobieństwo, że system A daje średnio lepszą klasyfikację niż system B jest nie mniejsze niż p. W przeciwnym przypadku udzielamy odpowiedzi negatywnej.*

Uwagi :

1. Algorytm stosować można tylko wówczas, kiedy uzasadnione jest założenie, że analizowane wyniki mają rozkład normalny.

2. Algorytm zakłada że każdy z wyników został wygenerowany niezależnie ze stosownego rozkładu. W rzeczywistych zastosowaniach to założenie jest zwykle nie do spełnienia, bo dysponujemy skończonym zbiorem danych, z którego generujemy zestawy zbiorów treningowych i testowych. W przypadku 10-krotnej kroswalidacji dla każdej pary przebiegów część wspólna zbiorów treningowych to $\frac{8}{9}$ każdego z tych zbiorów, więc nie można oczekiwać całkowitej niezależności wyników.

3. Najczęściej stosuje się parametr p o wartości 0.95, czyli dający spore prawdopodobieństwo istotności różnicy.

4. W praktycznej realizacji algorytm można nieco uprościć numerycznie, by np. uniknąć dzielenia i zaraz potem mnożenia przez tę samą wartość. W przedstawionej formie jest jednak bardziej zrozumiały niż w zoptymalizowanej.

5. Kiedy porównujemy systemy mając do dyspozycji średnie i wariancje (bez pełnych oryginalnych wyników), wówczas zaczynamy algorytm od punktu 2.

6. Punkt 2 ma na celu naniesienie stosownej poprawki do szacowanej wariancji. Zakłada, że oba zestawy danych pochodzą z tego samego rozkładu, co nie zawsze jest prawdą. Aby pozbyć się tego założenia można pominąć ten punkt algorytmu, a w kolejnym zamiast σ^2 użyć odpowiednio σ_A^2 oraz σ_B^2.

7. Jeśli porównywane systemy mogą być uruchomione dużą liczbę razy, bo działają wystarczająco szybko, to zamiast szacować wariancję średnich różnic tak jak to przedstawia punkt 3 algorytmu, można oszacować ją tak jak dla zmiennych A oraz B w punkcie 1.

Algorytm 2.1 stosuje się wówczas, gdy rozkłady wyników dla systemów A i B są niezależne. Takie założenie należy przyjmować, kiedy chcemy porównać systemy mając tylko średnie i odchylenia standardowe wyników. Podejście to ma jednak tę wadę, że może znacznie przeszacowywać wariancję różnicy średnich, ponieważ nie uwzględnia faktu, że na podobnych danych systemy będą dawały dość skorelowane rezultaty, co mimo sporej wariancji dla każdego z systemów może skutkować stosunkowo małą wariancją różnicy. Nadmiernie pesymistyczna ocena wariancji prowadzi do bardzo szerokich przedziałów ufności, co sprawia, że różnice między wynikami różnych systemów będą zbyt rzadko uznawane za statystycznie istotne. Zdecydowanie rzetelniejszym porównaniem metod jest zastosowanie ich do dokładnie tych samych zbiorów danych i użycie algorytmu 2.2 dla określenia statystycznej istotności różnic.

Algorytm 2.2 (Test t Studenta dla zmiennych skorelowanych)

► ***Dane:*** *Wyniki* $A_1, \ldots, A_n, B_1, \ldots, B_n \in [0,1]$, $n \in N$ *dla systemów klasyfikacji A oraz B (uzyskane odpowiednio dla tych samych zbiorów danych treningowych i testowych), poziom pewności odpowiedzi p.*

◄ ***Wynik:*** *Odpowiedź na pytanie czy system A daje z prawdopodobieństwem co najmniej p lepszą jakość klasyfikacji niż system B.*

1. *Dla zmiennej losowej $D = A - B$, na podstawie próbki $\{D_i = A_i - B_i : i = 1, \ldots, N\}$ szacujemy średnią i wariancję rozkładu:*

$$M_D = \frac{1}{n}\sum_{i=1}^{n} D_i \qquad \sigma_D^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} D_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} D_i\right)^2\right)$$

2. *Szacujemy wariancję zmiennej M_D:*

$$\sigma_M^2 = \frac{\sigma_D^2}{n}$$

3. *Wyznaczamy wartość*

$$t = \frac{M_D}{\sigma_M}$$

4. *Jeśli $t \geqslant t_{p,n-1}$, gdzie $t_{p,n-1}$ jest kwantylem dla prawdopodobieństwa p rozkładu t Studenta o $n-1$ stopniach swobody, to kończymy odpowiadając, że prawdopodobieństwo, że system A daje średnio lepszą klasyfikację niż system B jest nie mniejsze niż p. W przeciwnym przypadku udzielamy odpowiedzi negatywnej.*

Niestety, wymóg zebrania wyników poprzez zastosowanie porównywanych systemów do tych samych danych treningowych i testowych bardzo utrudnia użycie algorytmu 2.2. W szczególności nie można w ten sposób porównać własnych wyników z wynikami znalezionymi w publikacjach, jeśli nie dysponuje się programem, który można samemu użyć, a tylko otrzymanymi przez innych rezultatami.

Założenie, że rozkład wyników, których średnie liczymy, jest normalny oraz, że kolejne z uśrednianych wartości są niezależnymi losowaniami z tego rozkładu sprawia, że w zasadzie nie powinno się w ten sposób porównywać wyników testów, kiedy używamy jednoznacznego podziału na zbiory treningowy i testowy, ani kiedy wykonujemy test L1O. W przypadku L1O pojedynczy wynik testu jest wartością oczekiwaną rozkładu dwupunktowego a nie normalnego. W obu przypadkach wyniki uzyskane są na skutek trenowania systemu na takich samych próbkach danych, co w przypadku metod deterministycznych da jednopunktowy rozkład wyników, natomiast rozkład normalny tylko wówczas, gdy trenowany model jest losowy w taki sposób, że wielokrotnie trenowany na tym samym zbiorze danych wygeneruje wyniki o rozkładzie normalnym. Jeśli więc zbiór danych jest ograniczony (co w rzeczywistych zastosowaniach zawsze jest prawdą, bo tylko w sztucznie zdefiniowanych problemach może istnieć możliwość generowania dowolnej liczby niezależnych zestawów danych), to najbardziej wiarygodne wyniki porównania można uzyskać stosując test kroswalidacyjny.

Istnieją także alternatywne metody oceny statystycznej istotności różnic (np. test McNemar'a). Próbę porównania różnych metod, podjętą przez Dietterich'a, można znaleźć w [39, 40].

2.3 Indukcja reguł logicznych

Jedną z najbardziej zrozumiałych dla człowieka form opisów działania klasyfikatorów są reguły logiczne. Ten termin nie jest jednak bardzo precyzyjny – zawiera w sobie całą gamę różnych postaci reguł, bo reguły mogą być definiowane w różnych językach logiki. Różne też mogą być podejścia do generowania regułowych opisów danych – niektóre metody próbują opisywać wiedzę zgromadzoną przez systemy klasyfikacji, które nie są wyposażone w możliwość opisu swego działania, inne korzystają bezpośrednio z danych.

2.3.1 Różne formy reprezentacji wiedzy

Wiedzę odkrytą w danych można przedstawić na wiele różnych sposobów. Od tego, jakiej formy reprezentacji wiedzy potrzebujemy, będzie w dużym stopniu zależało jakiego typu systemów użyjemy. Najbardziej celowym wydaje się szukanie reguł logicznych opisujących dane. Istnieją różne typy reguł logicznych, a najbardziej zrozumiałe dla człowieka wydają się być reguły logiki klasycznej pierwszego rzędu. Reguły rozmyte [82, 85, 105, 139, 179] są rzadziej stosowane w zadaniach wydobywania wiedzy z danych, choć w pewnych sytuacjach użycie rozmytych zmiennych lingwistycznych może być nawet bardziej intuicyjne.

W niektórych przypadkach warto jest użyć reguł w formie **M-z-N** [177] tzn. takich, których przesłanki zawierają zdania typu „*M spośród N podanych warunków jest spełnionych*". Tego typu sformułowania, podobnie jak miary odległości, pozwalają wyrażać koncepcje dotyczące jednocześnie wielu warunków (na przykład w opisie wyników głosowania można w zwarty sposób wyrazić stwierdzenie „większość jest za"), które trudno wyrazić prostymi regułami logiki klasycznej. Reguły typu M-z-N potrafią znacznie uprościć zapis, przy czym pozostają dość łatwe w interpretacji.

Postać klasycznych reguł logicznych jest jedną z najprostszych form reprezentacji wiedzy. Jednak, w bardziej złożonych przypadkach, dobra klasyfikacja opierająca się wyłącznie na cechach opisujących obiekty w bazie danych może nie być możliwa – należy wówczas podejmować próby szukania przydatnych cech (np. przez liniowe lub nieliniowe transformacje dostępnych cech). Niektóre systemy mogą łatwiej znaleźć interesujące rozwiązania, kiedy zredukuje się liczbę cech stosując np. analizę czynników głównych lub niezależnych. Zakładamy tutaj, że problem opisany jest w wektorowej przestrzeni cech (np. zgodnie z defi-

nicjami wprowadzonymi w rozdziale 2) w taki sposób, że poddaje się klasyfikacji za pomocą sieci neuronowych lub innych systemów.

Problem generowania reguł logicznych z danych lub z sieci neuronowych rozwiązać można na wiele sposobów. Większość z nich nie daje możliwości kontrolowania zbioru reguł pod względem dokładności i czytelności. Użytecznym może być dysponowanie kilkoma zbiorami reguł: od najprostszych w formie i najbardziej ogólnych do bardziej szczegółowych i dokładnych. Dość istotną sprawą (zwłaszcza w zastosowaniach medycznych) jest również kwestia wiarygodności reguł, która zwykle może być osiągnięta kosztem ich dokładności.

Reguły logiczne są bardzo atrakcyjną formą reprezentacji wiedzy także z punktu widzenia zastosowań do wspomagania decyzji. Systemy ekspertowe [132, 140] budują swoje bazy wiedzy właśnie w oparciu o reguły wnioskowania.

2.3.2 Generowanie reguł na podstawie danych

Podstawowym narzędziem do indukcji reguł z danych są algorytmy *drzew decyzji*. Hierarchiczna struktura drzewa najczęściej pozwala na to by w sprawny sposób przekształcić ją do postaci reguł klasyfikacji, które dokładnie odzwierciedlają decyzje drzewa. Szereg najbardziej znanych metod z tej dziedziny przedstawia rozdział 2.9.

Istnieją także metody indukcji reguł bezpośrednio z danych. Rozdział 2.10 prezentuje pewne powszechnie znane i używane systemy indukcji reguł.

2.3.3 Generowanie reguł z nauczonych modeli

Bardzo szeroko rozwinęła się gałąź Inteligencji Obliczeniowej, która zajmuje się generowaniem reguł na podstawie gotowych (nauczonych) modeli. Uzyskane w ten sposób reguły nie opisują bezpośrednio danych, ale sposób podejmowania decyzji przez niezależne modele (zwykle sieci neuronowe). Rozwiązanie to posiada kilka zalet. Przede wszystkim uniezależnia od zbioru treningowego (który często jest dość ubogi), bowiem mając do dyspozycji klasyfikator można w każdej chwili wygenerować odpowiedni zestaw danych i nadać im etykiety klas (stąd używany klasyfikator nazywa się często „wyrocznią”). Inną zaletą jest fakt, że można w ten sposób wykorzystać możliwości przeróżnych metod (np. opartych na podobieństwie czy statystycznych), które same nie generują reguł, ale są w stanie dobrze nauczyć się danego problemu, więc w pewnych przypadkach takie dwuetapowe podejście do indukcji reguł może łatwiej przynieść owoce. Poważną wadą takich rozwiązań jest możliwość nałożenia się dwóch błędów – modelu uczącego się z danych i modelu tworzącego reguły, więc uzyskanie dużej dokładności reguł na surowych danych jest znacznie utrudnione. Z tego powodu bardziej uzasadnionym podejściem do celu generowania reguł logicznych na podstawie baz danych

wydaje się być modyfikowanie algorytmów uczenia systemów inteligencji obliczeniowej w taki sposób, by bezpośrednio po nauczeniu systemu móc z łatwością opisać jego działanie przez zbiór reguł logicznych.

Tickle i in. podjęli próbę opracowania taksonomii neuronowych algorytmów generowania reguł [175]. Zaproponowali charakteryzowanie metod ze względu na rodzaj generowanych reguł, ich jakość (dokładność klasyfikacji, liczbę, zwartość), czytelność reguł, złożoność algorytmu, czy sposób analizy sieci neuronowej. Do tej taksonomii warto jeszcze dodać sposób użycia zmiennych lingwistycznych, koniecznych do sformułowania reguł.

Jednym z dość efektywnych systemów generujących reguły logiczne opisujące wyrocznię jest *TREPAN* stworzony przez Cravena i Shavlika [35], który generuje drzewo decyzji w oparciu o analizę odpowiedzi sieci neuronowej dla przedstawionych jej próbek danych.

VIA (ang. *Validity Interval Analysis*) jest systemem opracowanym przez Thruna [174] operującym na przedziałach walidacyjnych, które przedstawiają zakresy maksymalnych wzbudzeń neuronów. Można ich szukać standardowymi metodami programowania liniowego, a także poprzez sieci neuronowe, konstruowane na wzór sieci uczonych algorytmem propagacji wstecznej błędu.

Towell i Shavlik [177] przedstawili algorytm uczący sieć neuronową tak, by łatwo było wygenerować z niej zestaw reguł typu M-z-N. Zbierają oni w grupy połączenia ze zbliżonymi do siebie wagami i zastępują te wagi średnimi wartościami dla całej grupy, eliminując przy tym niepotrzebne wagi. Każda z grup może być opisana przez jedną przesłankę typu M-z-N. Do tego typu metody dodać można stosowną modyfikację wag po porównaniu wektora wag z wektorami wzorcowymi (odpowiadającymi wzorcowym regułom) [124]. Metoda *RuleNet* [2] również wykorzystuje podobne wzorce i potrafi wyszukiwać najlepsze reguły typu M-z-N w $O(n^2)$ kroków i najlepsze zbiory zagnieżdżonych reguł w $O(n^3)$ kroków. Ta metoda operuje jednak tylko na danych dyskretnych, a więc cechy o wartościach ciągłych muszą być najpierw zdyskretyzowane.

Rule Extraction As Learning (REAL) jest ogólną techniką stopniowego budowania zestawu reguł zaprezentowaną przez Cravena i Shavlika [36]. Dla nowego przypadku, który jest błędnie klasyfikowany przez dotychczasowy zestaw reguł tworzy się nową regułę i sprawdza wierność powiększonego zestawu z odpowiedziami z sieci neuronowej. Na podobnej zasadzie działa system *RULENEG* [3, 147, 86].

W metodzie *BRAINNE* [162] sieć o m wejściach i n wyjściach jest przekształcana w sieć o $m+n$ wejściach i n wyjściach i ponownie trenowana. Wejścia dla których wagi nieco się zmieniają po restrukturyzacji sieci są najbardziej istotnymi i są wykorzystywane do budowania reguł.

Powyższe metody są przykładami metod globalnych tzn. analizujących jednocześnie wyjścia dla całej sieci i dla wszystkich próbek danych. Istnieją także

metody lokalne tzn. takie, które analizują fragmenty sieci (często pojedyncze neurony ukryte) w poszukiwaniu reguł opisujących ich zachowanie. Wykorzystywane tutaj sieci używają sigmoidalnych albo zlokalizowanych funkcji transferu. Reguły opisujące działanie całej sieci są tworzone jako stosowne kombinacje reguł odpowiadających poszczególnym węzłom.

Lokalne metody indukcji reguł były przedstawiane m.in. przez Lin Min Fu [66, 67, 68, 69] oraz Gallanta [70]. Podobnie jak w przypadku metod globalnych można tutaj ograniczać głębokość szukania (Sethi i Yoo [163]). Towell i Shavlik w algorytmie *Subset* używają heurystyki polegającej na analizowaniu wag w porządku malejącym, przez co najpierw znajduje się najbardziej ogólne reguły, a potem coraz bardziej szczegółowe. Hayashi [85] opracował wersję tej metody generującą reguły rozmyte.

Podejmowano także próby indukcji reguł logicznych poprzez samoorganizujące się modele typu ART [87] i rozmyte ARTMAP [172]. Te ostatnie dają dodatkowo współczynniki pewności dla reguł. Prostsze architektury samoorganizujące się były także używane do celów geneowania reguł [178], ale dawały raczej mierne wyniki w problemach klasyfikacyjnych.

Algorytm *DEDEC* [3, 176] generuje reguły szukając minimalnego zestawu cech wystarczającego z punktu widzenia sieci neuronowej do rozróżnienia zadanego wzorca od innych. Nowy zbiór danych treningowych jest generowany przez zastępowanie oryginalnych przypadków całymi grupami, a wejścia są uporządkowane wg ich wpływu na klasyfikację. Tylko najważniejsze wejścia biorą udział w tworzeniu reguł, znajdowanych metodami sprawdzania różnych kombinacji wejściowych.

2.3.4 Sieci neuronowe dla indukcji reguł

Rodzina systemów, które polegają na modyfikacji algorytmów uczenia w sposób ułatwiający generowanie reguł, jest również dość liczna.

Najprostszy przypadek sieci neuronowych interpretowalnych logicznie to sieci, w których sygnały wejściowe i wyjściowe wszystkich neuronów w sieci są binarne. Wówczas po wytrenowaniu sieci można z niej z łatwością „wyczytać" zestaw reguł: wystarczy sprawdzić wyjście dla wszystkich możliwych kombinacji wejść tworząc dla każdej z nich po jednej regule. Przy założeniu, że dana cecha może się pojawiać w regule wprost, pojawiać się zanegowana albo nie pojawiać się, dla n binarnych cech mamy do sprawdzenia 3^n różnych reguł. Ze wzrostem n to zadanie szybko może się okazać zbyt kosztowne obliczeniowo, więc często szuka się metod ograniczania przestrzeni poszukiwań. Można na przykład ograniczyć liczbę przesłanek, które mogą się pojawiać w regule. Saito i Nakano [158] ograniczają głębokość drzewa przeszukiwań i pozwalają tylko na takie kombinacje literałów, które występowały w danych treningowych. Wadę takiego rozwiąza-

nia, polegającą na akceptowaniu zbyt ogólnych reguł, wyeliminował Gallant [70], zawężając reguły przez dokładanie cech w nich nie występujących i sprawdzanie ich wszystkich możliwych wartości.

Setiono i Liu [164] używają członu regularyzacyjnego w funkcji kosztów dla eliminowania małych wag. Podobna idea przyświeca metodzie *Successive Regularization* opracowanej przez Ishikawę [94], gdzie kładzie się nacisk na to, by neurony ukryte były w pełni wzbudzone albo całkowicie nieaktywne, przy czym stosowny człon regularyzacyjny dba o eliminację wag mniejszych od pewnego progu (metodę nazwano mianem „selektywnego zapominania" – ang. *selective forgetting*).

Inną metodę należącą do tej grupy zastosowali Geczy i Usui [71]: wagi sieci typu MLP są tutaj po zakończeniu procesu uczenia przekształcane w 0, $+1$ lub -1 w celu ułatwienia szukania reguł.

Andrews i Geva stworzyli metodę *RULEX* [4] wykorzystującą sieci neuronowe typu MLP z liniowymi kombinacjami par funkcji sigmoidalnych, o niezerowych wartościach w przedziałach, z których potem można wprost wyczytać reguły.

2.4 Metody optymalizacji

Zawężenie poszukiwań z przestrzeni wszystkich możliwych klasyfikatorów do przestrzeni modeli o określonej strukturze ma na celu usprawnienie szukania optymalnego modelu. Nie oznacza to jednak, że możliwe jest pełne przeszukanie przestrzeni modeli. Wybór optymalnego zestawu parametrów pozostaje najczęściej bardzo złożonym problemem, a sposób jego rozwiązywania dla danego problemu klasyfikacji jest obok definicji samej struktury modelu kluczowym elementem systemu klasyfikacji.

Kiedy funkcja określająca błąd popełniany przez model jest różniczkowalna, wówczas najczęściej stosuje się gradientowe metody minimalizacji, które stosunkowo szybko potrafią zlokalizować minimum funkcji błędu, jednak często jest to minimum lokalne. W niektórych metodach stosuje się pewne techniki mające na celu zwiększenie prawdopodobieństwa znalezienia globalnego minimum. Istnieją nawet dowody na to, że przy pewnych założeniach dana metoda minimalizacji na pewno znajdzie globalne minimum, jednak w praktyce jest to na ogół nieosiągalne, ponieważ założenia dowodu przełożone na praktykę najczęściej wiążą się z bardzo długim (uwzględniając możliwości obliczeniowe można powiedzieć, że nieskończonym) czasem działania procesu minimalizacji.

Modele, które nie mają różniczkowalnej funkcji błędu nie mogą być odnajdywane w procesach gradientowej minimalizacji. W takich przypadkach stosuje się najczęściej metody szukania. Można w tym celu stosować metody globalne

tzn. takie, które przy pewnych założeniach dają gwarancję znalezienia globalnego optimum, jednak w praktyce potrzeba metod uproszczonych, które dadzą wyniki w zdecydowanie krótszym czasie. W praktyce zastosowanie metod globalnych nie daje gwarancji znalezienia globalnego optimum jako, że parametry sterujące przebiegiem procesu ustawia się tak, by algorytm działał w sensownie krótkim czasie, a to oznacza rezygnację z gwarancji globalności znajdywanych rozwiązań.

Dla przyspieszenia procesów szukania można więc z równie dobrym skutkiem stosować heurystyki zawężające obszary poszukiwań. Metody heurystyczne zwykle zdecydowanie przyspieszają proces, ale nie dają gwarancji optymalności rozwiązania. Jednakże cel globalnej optymalności najczęściej schodzi na plan dalszy z powodu nierealności jego osiągnięcia w akceptowalnym czasie. Bardzo często godzimy się również z tym, że proces minimalizacji używa funkcji błędu zdecydowanie różnej od tej, która jest rzeczywistym celem. Na przykład metody gradientowe, które wymagają m.in. ciągłości funkcji błędu minimalizują błąd średniokwadratowy, który jest bardziej odpowiedni dla problemów aproksymacji niż klasyfikacji, w związku z czym znalezione minimum nie gwarantuje nam minimalności błędu w ocenie klasyfikacji.

Zarówno metody gradientowe, jak i metody szukania pozwalają osiągać bardzo interesujące rezultaty. Nie istnieje metoda, którą można by było obiektywnie uznać za najlepszą, dlatego najsłuszniejszym podejściem do rozwiązywania problemów klasyfikacji jest zastosowanie do danego zadania kilku różnych metod i ocena, który z systemów daje najbardziej atrakcyjne rozwiązania.

Interesujące zestawienie różnych metod globalnej optymalizacji przedstawionych w zwarty sposób można znaleźć w [56]. Szczegółowe opisy kilku gradientowych metod przedstawione zostały m.in. w [9, 57]. Poniższe podrozdziały przedstawiają tylko pobieżny przegląd różnych metod.

2.4.1 Metody gradientowe

Metody gradientowe polegają na wykorzystaniu wektora pochodnych cząstkowych funkcji błędu w celu zmniejszania wartości tej funkcji poprzez odpowiednią modyfikację parametrów.

Najprostszą metodą gradientowej minimalizacji jest metoda *najbardziej stromego (największego) spadku* (ang. *steepest descent*) funkcji błędu. Polega ona na zmianie parametrów w kierunku ujemnego gradientu (bardziej poprawnie: wektora przeciwnego do gradientu) w punkcie wyznaczonym przez aktualne wartości parametrów adaptacyjnych. A zatem jeśli $\mathbf{w}$ jest wektorem parametrów modelu to przyrost wartości parametrów w kroku t jest określony przez:

$$\Delta \mathbf{w}^{(t)} = -\eta \nabla E|_{w^{(t)}}, \tag{2.9}$$

gdzie η jest tzw. parametrem uczenia. Ponieważ ujemny gradient wskazuje kierunek największego spadku funkcji błędu, a nie kierunek do minimum tej funkcji, więc przebieg takiej minimalizacji może charakteryzować się dużymi oscylacjami, które powoli będą zbliżały wektor parametrów do optymalnej wartości. Dla uniknięcia tych oscylacji (a co za tym idzie przyspieszenia procesu) stosuje się tzw. uczenie z *momentem*, które polega na zmianie parametrów uzwględniającej nie tylko aktualną wartość gradientu, ale również przyrost wyznaczony w poprzedniej iteracji:

$$\Delta \mathbf{w}^{(t)} = -\eta \nabla E|_{w^{(t)}} + \mu \Delta \mathbf{w}^{(t-1)}. \tag{2.10}$$

Parametry η oraz μ mają istotny wpływ na przebieg procesu minimalizacji. Ich optymalne wartości są różne dla różnych problemów a nawet dla różnych etapów uczenia tego samego problemu. Istnieją metody usiłujące automatycznie dobierać te wartości odpowiednio do aktualnej sytuacji.

Optymalne wartości parametrów uczenia to takie, które powodują taką zmianę wartości parametrów, że wartość funkcji błędu jest najniższa z możliwych. Innymi słowy chodzi o to, by odnaleźć minimum funkcji błędu dla kierunku wyznaczonego gradientem. Można to wykonać na wiele sposobów – jeden z bardzo efektywnych jest oparty na wyznaczaniu minimum paraboli interpolującej funkcję błędu wzdłuż tego kierunku.

Znalezienie minimum wzdłuż kierunku wskazanego gradientem powoduje, że w następnym kroku gradient będzie prostopadły do poprzedniego, co oznacza, że do lokalnego minimum zmierzamy nie wprost, a oscylacjami podobnie jak przy zastosowaniu najprostszej metody największego spadku.

Rozwiązaniem tego problemu jest zastosowanie metody tzw. *sprzężonych gradientów*. Gradient jest sprzężony z poprzednim, o ile jego komponent równoległy do poprzedniego pozostaje zerowy (jak najbliższy zeru), czyli przenosi nas do miejsc, które pozostają minimami wzdłuż poprzedniego kierunku. W każdym kolejnym kroku szukamy gradientu sprzężonego z każdym z dotychczasowych. Dla kwadratowych funkcji błędu strategia ta zapewnia dotarcie do minimum w *n* krokach, gdzie *n* to wymiar przestrzeni (liczba parametrów).

Inną interesującą metodą jest metoda *Levenberga-Marquardta*, która została opracowana specjalnie dla funkcji błędu liczonych jako sumy kwadratów „mniejszych" błędów (w praktyce: różnic pomiędzy wartościami oryginalnymi a wynikami aproksymacji).

Bardzo popularną metodą gradientową jest (stosowana w sieciach neuronowych) metoda propagacji wstecznej błędu opisana w rozdziale 2.8.1.

2.4.2 Algorytmy szukania

Przeszukiwanie przestrzeni modeli metodą *generowania* po kolei wszystkich możliwych rozwiązań i *testowania* ich jakości, zazwyczaj nie daje się zastosować w praktyce, ze względu na dużą złożoność obliczeniową. Kiedy przestrzeń rozwiązań daje się uporządkować w strukturę drzewa, można w celu szukania modeli stosować metody przeszukiwania drzew i grafów. Najprostsze metody takie jak szukanie *w głąb* i *wszerz* podobnie jak metoda generowania i sprawdzania mogą być użyte tylko w przypadku takich definicji modeli, które wyznaczają bardzo małą przestrzeń. Dużo praktyczniejsze są metody heurystycznego szukania, które pozwalają szybko dochodzić do interesujących rozwiązań, również wtedy, gdy pełne przeszukanie całej przestrzeni nie jest możliwe.

Często stosowaną metodą jest metoda *wspinaczki* (ang. *hill climbing*), która różni się od przeszukiwania w głąb tylko tym, że przeszukuje kolejne poddrzewa w kolejności malejącej wartości zadanej heurystycznej oceny jakości. Podobnie działa również metoda „*najpierw najlepszy*", która zapamiętuje wszystkie stany odwiedzone do danej chwili i przeszukuje kandydujące poddrzewa (nie tylko poddrzewa aktualnego stanu, ale wszystkie odwiedzone) w kolejności malejącej wartości heurystyki. W szczególnych przypadkach, kiedy każdy nowy stan zapewnia wartość heurystyki wyższą niż jego rodzic, obie metody są tożsame.

Dokładniejszą (więc bardziej wymagającą obliczeniowo) jest metoda szukania *wiązką* (ang. *beam search*). Polega ona na przeszukiwaniu z zapamiętywaniem po każdym kroku w najlepszych rozwiązań ($w \in N$ nazywamy szerokością wiązki). Każdy kolejny krok polega na wygenerowaniu dla wszystkich stanów należących do wiązki wszystkich stanów potomnych i wyborze spośród nich kolejnej wiązki składającej się z w najlepszych. Dość dokładne studium różnych metod szukania prezentuje Winston w [184].

Bardzo skuteczną metodą szukania, która nie wymaga drzewiastej struktury przestrzeni modeli, a tylko umiejętności losowego generowania nowych modeli jest metoda *symulowanego wyżarzania* (ang. *simulated annealing*) zwana też *stopniowym schładzaniem*, którą po raz pierwszy przedstawili w 1983 roku Kirkpatrick, Gellat i Vecchi [108]. Jest to metoda inspirowana zjawiskami fizycznymi, która jest sterowana parametrem zwanym (na skutek fizycznych korzeni) *temperaturą*. Polega na generowaniu kolejnych rozwiązań w okolicy bieżącego stanu (dla wysokiej temperatury okolicą jest bardzo duży obszar – nawet obejmujący całą przestrzeń) i ocenianiu ich funkcją błędu – to czy zostaną zaakceptowane zależy od wielkości błędu oraz temperatury (dla wysokich temperatur zaakceptowane może być również rozwiązanie dające wyższy błąd od dotychczasowego). Algorytm jest więc określony przez trzy funkcje:

- określającą wartość temperatury w kolejnych krokach, np.:

$$T(k) = \frac{T_0}{\ln k}, \tag{2.11}$$

gdzie T_0 to odpowiednio duża wartość początkowa.

- gęstość prawdopodobieństwa zmian wartości parametrów $x_1, \ldots, x_D$:

$$g(\Delta x) = (2\pi T)^{-D/2} e^{\frac{-\Delta x^2}{2T}}, \tag{2.12}$$

- dystrybuantę rozkładu zmiennej losowej decydującej o akceptacji nowych stanów:

$$h(\Delta E) = \frac{1}{1 + e^{\frac{\Delta E}{T}}}. \tag{2.13}$$

Algorytm stopniowego schładzania doczekał się wielu modyfikacji, z których największy sukces osiągnęła metoda *adaptacyjnego symulowanego wyżarzania* opracowana przez Ingbera (ang. *Adaptive Simulated Annealing* – ASA) [91, 92, 93], która początkowo nosiła nazwę *Very Fast Simulated Re-annealing* – VFSR. Modyfikacje wprowadzone do oryginalnej wersji algorytmu stopniowego schładzania to:

- stosowanie niezależnych rozkładów zmian dla różnych parametrów,
- inna (sparametryzowana dla zwiększenia możliwości dopasowania do różnych problemów) funkcja określająca wartości temperatury. Szybkie i powolne schematy studzenia.

2.4.3 Programowanie matematyczne

Systemy klasyfikacji wykorzystują także metody programowania matematycznego (zarówno liniowego jak i nieliniowego).

W problemach, które dadzą się zdefiniować jako zadania minimalizacji bądź maksymalizacji liniowej kombinacji zmiennych przy ograniczeniach w postaci liniowych równań bądź nierówności, można skorzystać z najpopularniejszego algorytmu programowania liniowego, czyli metody „*sympleks*”. Wykorzystuje ona fakt, że wystarczy ograniczyć szukanie optymalnych rozwiązań do tzw. *dopuszczalnych rozwiązań bazowych*, by iteracyjnie generując szereg rozwiązań bazowych doprowadzić do określenia optymalnego wektora bądź stwierdzenia nierozwiązywalności zadania. Szczegóły na temat algorytmu „sympleks” można znaleźć np. w [156, 146], a szczegółową analizę algorytmu w [170]. Algorytm jest bardzo popularny i szeroko stosowany pomimo niewielomianowej złożoności. Istnieją

również metody o złożoności wielomianowej (np. metoda *Chacziana*), jednak mimo to są one mało użyteczne w praktyce.

Również metody programowania nieliniowego [157] znajdują zastosowania w systemach klasyfikacji. Spektakularnym przykładem są modele *Support Vector Machines* (SVM – p. rozdział 2.7), w których zadanie minimalizacji wartości funkcji błędu jest sformułowane jako zadanie programowania kwadratowego.

2.5 Naiwny Klasyfikator Bayesowski

Choć twierdzenie Bayesa pozwala sformułować optymalny klasyfikator (p. rozdział 2.2.2), to zastosowania w praktyce mogą znaleźć jedynie znacznie uproszczone metody. Jednym z takich uproszczeń jest klasyfikator Bayesowski zwany *naiwnym* [131]. Jego naiwność polega na założeniu, że zmienne losowe odpowiadające poszczególnym wymiarom przestrzeni są niezależne, co znacznie ułatwia wyliczenie prawdopodobieństw przynależności do poszczególnych klas, więc i wyznaczenie klasy maksymalizującej prawdopodobieństwo *a posteriori*:

$$NKB(x) = \operatorname*{argmax}_{c \in C} P(c|x) = \operatorname*{argmax}_{c \in C} \frac{P(x|c)P(c)}{P(x)}$$

$$= \operatorname*{argmax}_{c \in C} P(c) \prod_i P(x_i|c) \tag{2.14}$$

W przypadku przestrzeni cech dyskretnych prawdopodobieństwa $P(x_i|c)$ mogą być z łatwością wyliczone na podstawie danych treningowych.

Najczęściej Naiwny Klasyfikator Bayesowski jest stosowany właśnie dla danych dyskretnych (jeśli pewne wymiary przestrzeni są ciągłe, to poddaje się je dyskretyzacji). Można jednak szacować prawdopodobieństwa *a priori* dla cech ciągłych zakładając normalność rozkładu każdej z cech dla każdej z klas. Wówczas bez uszczerbku dla procesu maksymalizacji wystarczy przyjąć

$$P(x_i|c) = G(x, \mu_i^c, \sigma_i^c), \tag{2.15}$$

gdzie G jest gęstością rozkładu normalnego, a μ_i^c oraz σ_i^c to odpowiednio wartość średnia i odchylenie standardowe i-tej cechy dla klasy c, obserwowane w zbiorze treningowym [159].

W praktyce założenia co do niezależności rozkładów poszczególnych cech oraz normalności stosownych rozkładów mogą być dość dalekie od prawdy, a wówczas Naiwny Klasyfikator Bayesowski jest również daleki od optymalnego.

Istnieją również inne podejścia do problemu oszacowania prawdopodobieństw warunkowych dla cech ciągłych, które nie są obciążone założeniami normalności rozkładów [99].

2.6 Liniowa dyskryminacja

Zadanie liniowej dyskryminacji klas C_1 i C_2 polega na znalezieniu hiperpłaszczyzny, która w jak najlepszy sposób rozdziela dane należące do tych dwóch klas. W literaturze można znaleźć wiele opisów tego zagadnienia i różnych do niego podejść [159, 37].

Jedną z popularniejszych metod zaproponował Fisher w [62]. Polega ona na szukaniu projekcji przestrzeni na prostą (bez utraty ogólności można rozważać tylko proste przechodzące przez początek układu współrzędnych), która pozwoli separować klasy. Dla danego zbioru treningowego $T = \{(x_1, c_1), \ldots, (x_n, c_n)\} \subseteq X \times C$, gdzie $C = \{C_1, C_2\}$ celem jest więc znalezienie wektora w wyznaczającego projekcję

$$y_i = w^T x_i = < w, x_i > \quad i = 1, \ldots, n \tag{2.16}$$

separującą wartości dla wektorów różnych klas. Miarą separowalności, którą przyjął Fisher jest

$$J(w) = \frac{(\mu_{Y_1} - \mu_{Y_2})^2}{\sigma_{Y_1}^2 + \sigma_{Y_2}^2}, \tag{2.17}$$

gdzie μ_{Y_1} i μ_{Y_2} to wartości oczekiwane, a σ_{Y_1} i σ_{Y_2} to odchylenia standardowe rzutów wektorów z klas (odpowiednio) C_1 i C_2 (zbiory tych rzutów zostały oznaczone symbolami Y_1 oraz Y_2). Szacowanie tych wartości na podstawie zbioru treningowego T oraz uproszczenie pewnych stałych prowadzą do oceny:

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}, \tag{2.18}$$

dla

$$m_i = \frac{1}{|Y_i|} \sum_{y \in Y_i} x \qquad s_i = \sum_{y \in Y_i} (y - m_i)^2. \tag{2.19}$$

Istnieją różne metody wyznaczania wektora w_{opt} maksymalizującego wartość $J(w)$. Jedna z nich [159] prowadzi do określenia:

$$w_{\mathsf{opt}} = \frac{1}{2} k (\Sigma_1 + \Sigma_2)^{-1} (\mu_1 - \mu_2), \tag{2.20}$$

gdzie

$$k = \frac{\sigma_{Y_1}^2 + \sigma_{Y_2}^2}{\mu_{Y_1} - \mu_{Y_2}} \tag{2.21}$$

a μ_i oraz Σ_i są odpowiednio wartością oczekiwaną oraz macierzą kowariancji dla wektorów z klasy C_i. Czynnik skalarny $\frac{1}{2}k$ nie ma wpływu na rozwiązanie, bo zmienia tylko długość wektora w_{opt}, który wyznacza kierunek prostej.

2.7 Support Vector Machines

System przedstawiony przez Vapnika pod nazwą *Support Vector Machines* (SVM) [181] szuka hiperpłaszczyzny, która rozdziela wektory z dwóch klas, a dodatkowo zachowuje jak największy margines, czyli odległość od tej hiperpłaszczyzny do najbliższych wektorów zbioru treningowego. Jest więc pod tym względem bardzo zbliżony do metod liniowej dyskryminacji, jednakże SVM separuje liniowo nie oryginalne wektory, a ich obrazy w całkowicie innej przestrzeni, dzięki czemu jego granice decyzyjne mogą przybierać różne (nieliniowe) kształty.

Dla danego zbioru treningowego $T = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq R^k \times \{C_1, C_2\}$, poprzez (zazwyczaj nieliniową) transformację

$$\Phi : R^k \to \mathscr{F} \tag{2.22}$$

do przestrzeni $\mathscr{F}$ (zwykle dużej wymiarowości) otrzymujemy nowe zadanie klasyfikacji

$$T' = \{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_n), y_n)\} \subseteq \mathscr{F} \times \{-1, 1\}, \tag{2.23}$$

gdzie

$$y_i = \begin{cases} 1 & \text{o ile } c_i = C_1 \\ -1 & \text{w przeciwnym przypadku.} \end{cases} \tag{2.24}$$

Przy założeniu separowalności klas, celem jest znalezienie takiego wektora $\mathbf{w}$, dla którego

$$y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geqslant 1, \quad i = 1, \ldots, n. \tag{2.25}$$

Ponieważ rozwiązania różniące się tylko skalowaniem są z punktu widzenia separowania klas hiperpłaszczyzną tożsame, można narzucić ograniczenie

$$\tau ||\mathbf{w}|| = 1, \tag{2.26}$$

gdzie τ jest marginesem. Wówczas maksymalizację marginesu można uzyskać poprzez minimalizację normy wektora $\mathbf{w}$.

Aby metoda mogła być stosowana również dla nieseparowalnych danych warunki 2.25 zastępujemy następującymi:

$$y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \quad i = 1, \ldots, n. \tag{2.27}$$

Liczba niezerowych ξ_i jest wówczas liczbą błędów popełnianych w klasyfikacji wektorów zbioru treningowego. Zatem minimalizację liczby błędów możemy uzyskać np. minimalizując $\sum_{i=1}^{n} \xi_i$.

Szukanie optymalnego modelu SVM można więc zdefiniować jako problem minimalizacji

$$\min_{\mathbf{w}, b, \xi} \left[\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right] \tag{2.28}$$

przy ograniczeniach 2.27. C jest stałą – parametrem metody pozwalającym regulować pomiędzy dokładnością modelu a jego złożonością.

Odpowiednie przekształcenia prowadzą do dualnego problemu maksymalizacji

$$\max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right] \tag{2.29}$$

przy ograniczeniach

$$0 \leqslant \alpha_i \leqslant C, \qquad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0,$$

gdzie $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_k)$ są tzw. *funkcjami jądrowymi* (ang. *kernel functions*) systemu. W praktyce nie stosuje się jawnej transformacji Φ, tylko wylicza wprost wartości funkcji $K(\mathbf{x}, \mathbf{y})$ (co określa się nazwą *kernel trick*). Najbardziej popularną i najsprawniej działającą postacią tych funkcji są funkcje gaussowskie:

$$K(\mathbf{x}, \mathbf{y}) = \exp(-b||\mathbf{x} - \mathbf{y}||^2), \tag{2.30}$$

choć możliwe jest również stosowanie funkcji sigmoidalnych, wielomianowych i wielu innych.

Problem maksymalizacji 2.29 jest poprawnie zdefiniowanym problemem programowania kwadratowego. Jego rozwiązanie jest zadaniem bardzo złożonym, więc opracowano wiele usprawnień, z których jednym z ciekawszych jest algorytm *Sequential Minimal Optimization* (SMO) zdefiniowany przez Platta [145], który zamiast jednego skomplikowanego problemu rozwiązuje sekwencyjnie wiele problemów maksymalizacji dla odpowiednio wybranych par zmiennych.

2.8 Sieci neuronowe

Sieci neuronowe są systemami, które zrodziły się z inspiracji biologicznych. Ich podstawową ideą jest budowanie struktur złożonych z elementów zwanych *neuronami*, które mogą przesyłać między sobą sygnały. Każdy neuron jest jednostką przetwarzającą sumę sygnałów wejściowych (uwzględniającą *wagi* związane z połączeniami międzyneuronowymi) poprzez tzw. *funkcję aktywacji*, której wartość określa stopień wzbudzenia neuronu.

Istnieje wiele różnych architektur sieci neuronowych (różne liczby neuronów, różne schematy połączeń między neuronami), wiele różnych funkcji aktywacji (bogate zestawienie różnych funkcji można znaleźć w [55]), a także wiele różnych metod adaptacji parametrów, które mają na celu osiągnięcie jak najlepszej sieci dla danego problemu.

Sieci neuronowe znajdują zastosowanie w problemach klasyfikacji, ale także w zadaniach aproksymacji funkcji.

Jednym z popularniejszych typów sieci neuronowych jest sieć zwana *wielowarstwowym perceptronem* (ang. *Multi-Layer Perceptron* – MLP). Jest to sieć składająca się z pewnej liczby warstw neuronów (w każdej warstwie znajduje się pewna liczba neuronów), w której połączenia między neuronami mają miejsce tylko pomiędzy neuronami sąsiednich warstw i tylko w jedną stronę (ang. *feed-forward network*). Ostatnią z warstw nazywa się *warstwą wyjściową* a wszystkie wewnętrzne *warstwami ukrytymi*. W pewnych analizach wygodnie jest przyjąć istnienie warstwy neuronów wejściowych, realizujących identycznościową funkcję aktywacji. Dla każdego neuronu (nadajmy mu indeks j) warstw ukrytych i wyjściowej jego wzbudzenie wyraża się wzorem

$$z_j = g(a_j), \tag{2.31}$$

gdzie g jest funkcją aktywacji tego neuronu, oraz

$$a_j = \sum_i w_{ji} z_i. \tag{2.32}$$

Ważona suma z równania 2.32 uwzględnia wzbudzenie z_i każdego neuronu (o indeksie i), którego sygnał jest wysyłany do neuronu o indeksie j. Często suma ta jest wzbogacona o dodatkowy składnik b o angielskiej nazwie *bias*, jednak w związku z tym, że ten sam efekt można osiągnąć dodając neuron o stały wzbudzeniu równym 1 i łącząc go z neuronami ważonymi połączeniami, można go pominąć bez utraty ogólności rozważań.

Błąd popełniany przez sieć jest liczony jako suma błędów dla kolejnych wektorów zbioru testującego:

$$E = \sum_n E^n, \tag{2.33}$$

gdzie E^n jest funkcją oceniającą błąd dla danego wektora na podstawie wartości wzbudzeń neuronów warstwy wyjściowej:

$$E^n = E^n(y_1, \ldots, y_c). \tag{2.34}$$

2.8.1 Algorytm propagacji wstecznej błędu

Proces adaptacji parametrów sieci MLP bardzo często przebiega według *algorytmu propagacji wstecznej błędu*. Jest to ogólny schemat uczenia sieci, który nie zakłada żadnej konkretnej postaci funkcji aktywacji 2.31 ani funkcji błędu 2.34, a tylko ich różniczkowalność.

Algorytm propagacji wstecznej błędu jest metodą gradientową – każda waga w_{ji} dla danego błędu E^n będzie podlegać modyfikacji o

$$\Delta w_{ji} = -\eta \frac{\partial E^n}{\partial w_{ji}} \tag{2.35}$$

gdzie η jest parametrem uczenia. Ponieważ w_{ji} nie jest bezpośrednim parametrem funkcji błędu, korzystamy z reguły łańcuchowej dla pochodnych cząstkowych i otrzymujemy:

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i, \tag{2.36}$$

gdzie

$$\delta_j = \frac{\partial E^n}{\partial a_j}. \tag{2.37}$$

Dla neuronów warstwy wyjściowej (dla uwidocznienia wyjść sieci zastępujemy z_j przez y_j) otrzymujemy więc:

$$\delta_j = \frac{\partial E^n}{\partial a_j} = g'(a_j)\frac{\partial E^n}{\partial y_j}. \tag{2.38}$$

Dla neuronów warstw ukrytych korzystamy z reguły łańcuchowej otrzymując:

$$\delta_j = \frac{\partial E^n}{\partial a_j} = \sum_k \frac{\partial E^n}{\partial a_k}\frac{\partial a_k}{\partial a_j}, \tag{2.39}$$

gdzie indeks k przebiega po wszystkich indeksach neuronów, którym neuron j przekazuje swoje wyjście. Otrzymujemy więc:

$$\delta_j = g'(a_j)\sum_k w_{kj}\delta_k. \tag{2.40}$$

A zatem wyliczanie wartości współczynników δ_j odbywa się od warstwy wyjściowej (równanie 2.38) poprzez warstwy ukryte (równanie 2.40) propagując wyliczone wartości wstecz do poprzednich warstw (stąd nazwa algorytmu, jako że δ_j są nazywane *błędami*).

2.9 Drzewa decyzji

W problemach klasyfikacyjnych, w których ważna jest nie tylko poprawność klasyfikacji, ale również możliwość logicznej oceny proponowanych przez system rozwiązań, bardzo przydatne są tzw. *drzewa decyzji* (*drzewa klasyfikacji*).

Definicja 2.7 ***Węzłem*** *drzewa klasyfikacji dla przestrzeni klasyfikacji X i zbioru klas C nazywamy dowolną parę $W = (f,c)$, gdzie $f : X \to \{0,1\}$ oraz $c \in C$. Funkcję f nazywamy wówczas* ***funkcją przynależności*** *do węzła W, a klasę c* ***etykietą węzła*** *W.*

Uwaga : Ponieważ f jest funkcją logiczną, często jej przeciwdziedzinę określa się równoważnie jako $\{\text{fałsz}, \text{prawda}\}$.

Definicja 2.8 ***Drzewem klasyfikacji*** *dla przestrzeni klasyfikacji X i zbioru klas C nazywamy parę D spełniającą jeden z poniższych warunków:*

1° *$D = (W, ())$, gdzie W jest węzłem a $()$ jest pustą listą drzew.*

2° *$D = (W, (D_1, \ldots, D_n))$, gdzie $W = (f,c)$ jest węzłem, $n \in N$, dla każdego $i \in \{1, \ldots n\}$, $D_i = ((f_i, c_i), L_i)$ jest drzewem (które nazywamy* ***poddrzewem bezpośrednim*** *drzewa D) oraz dla każdego $x \in X$, $\sum_{i=1}^{n} f_i(x) \leqslant 1$ (tzn., że każdy wektor wpada co najwyżej do jednego podwęzła).*

Węzeł W nazywamy wówczas ***węzłem głównym*** *lub* ***korzeniem*** *drzewa D. Węzły główne drzew $D_1, \ldots, D_n$ nazywamy* ***podwęzłami*** *albo* ***dziećmi*** *węzła W, a węzeł W ich* ***nadwęzłem*** *lub* ***rodzicem****.*

Definicja 2.9 (Terminy pomocnicze)

Węzły mające wspólnego rodzica nazywamy ***rodzeństwem*** *(każdy węzeł jest dla pozostałych* ***bratem****).*

Poddrzewem *drzewa W jest jego każde poddrzewo bezpośrednie, a także każde poddrzewo dowolnego poddrzewa bezpośredniego.*

Węzłem drzewa klasyfikacji *D nazywamy węzeł główny drzewa, a także każdy węzeł dowolnego poddrzewa bezpośredniego D.*

Liściem *drzewa D nazywamy każdy węzeł drzewa W, który jest węzłem głównym poddrzewa z pustą listą poddrzew właściwych.*

Gałęzią *drzewa D nazywamy dowolny ciąg węzłów $(W_1, \ldots W_n)$ taki, że $n \in N$, W_1 jest węzłem głównym drzewa D, W_n jego liściem oraz dla każdego $i \in \{2, \ldots, n\}$ W_i jest podwęzłem węzła W_{i-1}.*

Długością gałęzi *nazywamy liczbę węzłów ją stanowiących.*

Głębokością drzewa *nazywamy maksymalną długość gałęzi tego drzewa.*

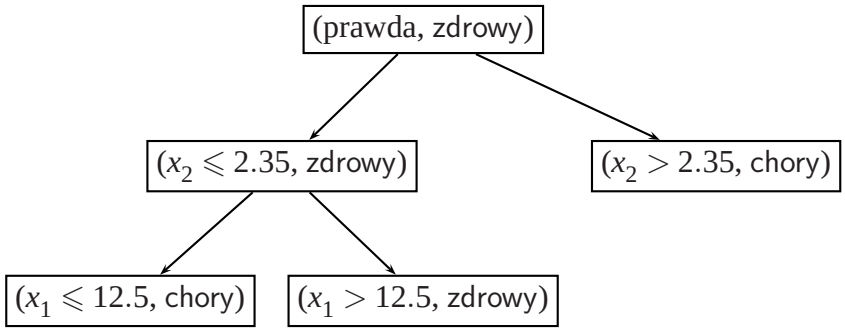
Drzewem binarnym *nazywamy drzewo, w którym każdy węzeł nie będący liściem posiada dokładnie dwa podwęzły.*

Mówimy, że $x \in X$ ***należy*** *(bądź* ***wpada****) do węzła $W = (f,c)$ drzewa D o ile $f(x) = 1$ oraz albo W jest węzłem głównym D, albo x należy do nadwęzła węzła W w drzewie D.*

Funkcja klasyfikująca odpowiadająca drzewu *D, to klasyfikator, który każdemu wektorowi $x \in X$ przyporządkowuje klasę, która jest etykietą liścia, do którego wpada x.*

Uwaga : W praktyce interesują nas tylko drzewa, w których funkcja węzła głównego jest stała o wartości 1, czyli drzewa dzielące całą przestrzeń klasyfikacji.

Przykład 2.4 *Niech $X = R \times \{a,b,c,d\}$ oraz $C = \{\mathsf{zdrowy}, \mathsf{chory}\}$. Diagram z rysunku 2.5 przedstawia przykład binarnego drzewa klasyfikacji (strzałki reprezentują relację bycia podwęzłem).*



Rysunek 2.5: Przykład drzewa klasyfikacji

Drzewa decyzyjne mogą być tworzone na podstawie wiedzy ekspertów, jeśli tylko ich wiedza przekłada się w prosty sposób na warunki logiczne określające poszczególne klasy. W praktyce jednak eksperci do podjęcia decyzji wykorzystują bardzo różne przesłanki (czasem nawet kierowani są intuicją zawodową) i często nie są w stanie zwerbalizować kryteriów, które zaważyły o ich decyzji. Algorytmy automatycznego tworzenia drzew decyzyjnych na podstawie sklasyfikowanych danych są często w stanie odtworzyć sposób myślenia eksperta albo też wykryć konsekwencje nieodpowiedniego przetwarzania danych[2]. Budowanie

[2]Nasz zespół spotkał się kilka lat temu z przypadkiem danych medycznych, przetworzonych ręcznie w sposób, który bardzo mocno wpływał na wyniki różnych systemów klasyfikacji. Dzięki zastosowaniu systemu, którego struktura była interpretowalna dla człowieka [50], udało się tę nieprawidłowość wykryć (dane były niekompletne i brakujące wartości zastąpiono średnimi wartościami dla poszczególnych cech, a jako, że braki te nie były przypadkowe mocno korelowały się z jedną z klas, co było wykorzystywane przez wszystkie systemy).

drzew decyzji polega zwykle na tym, że rozpoczynamy od drzewa złożonego z jednego liścia, w który wpadają wszystkie wektory przestrzeni, a w kolejnych etapach zamieniamy liście na poddrzewa, które lepiej rozdzielają różne klasy. Różne algorytmy budowania drzew decyzji różnią się między sobą sposobem szukania podziałów przestrzeni dających optymalne drzewa decyzji.

Klasyfikacja nowego wektora danych polega na znalezieniu takiej gałęzi drzewa, że wektor ten wpada do każdego z jej węzłów. Wówczas klasa przypisywana temu wektorowi, to klasa będąca etykietą liścia znajdującego się na tej gałęzi. Śledząc krok po kroku taką gałąź można analizować problem klasyfikacji stopniowo – od bardzo ogólnych decyzji (w okolicach korzenia) do bardzo szczegółowych (bliżej liści). Cecha ta sprawia, że każde drzewo klasyfikacji może być traktowane jako system wieloetapowy [117]. Jeśli problem jest wieloklasowy i eksperci są w stanie wskazać pewne grupy klas podobnych do siebie i rozgraniczać klasy na różnych poziomach (od bardzo ogólnego podziału do bardzo szczegółowego), to informacje takie mogą być również odpowiednio wykorzystane przy budowaniu stosownego drzewa.

Ważną zaletą drzew decyzji jest fakt, że ich tworzenie przebiega rekurencyjnie i w każdym kolejnym etapie zawężamy analizowany obszar przestrzeni, dzięki czemu istnieje możliwość wychwycenia tych cech, które są istotne lokalnie, choć niewiele są w stanie powiedzieć na temat klasyfikacji całej przestrzeni.

Specyfika konstrukcji drzew decyzji sprawia, że budowanie optymalnych struktur jest bardzo trudne. Jak pokazuje de Sa [38] maksymalizacja prawdopodobieństwa poprawności klasyfikacji dla danego węzła jest czymś innym niż maksymalizacja prawdopodobieństwa poprawności klasyfikacji całego drzewa. Dowód de Sa dotyczy drzew o specyficznej strukturze – takich, które mają po jednym liściu dla każdej klasy. W takim przypadku można każdemu z węzłów drzewa przypisać jako etykietę pewien podzbiór zbioru klas tak, by korzeń drzewa miał w etykiecie pełny zbiór klas, a kolejne podziały dzieliły zbiór klas danego węzła na rozłączne podzbiory. Przy założeniu niezależności cech wykorzystanych w drzewie można stwierdzić, że prawdopodobieństwo poprawności klasyfikacji do danej klasy wyraża się wówczas wzorem

$$P_c(k) = \prod_{i=1}^{n-1} P_c(k|W_i), \tag{2.41}$$

gdzie $(W_1, \ldots, W_n)$ jest gałęzią drzewa D o liściu z etykietą klasy k oraz $P_c(k|W_i)$ jest prawdopodobieństwem poprawnej decyzji na poziomie węzła W_i. Wówczas prawdopodobieństwo poprawności klasyfikacji całego drzewa określa wzór

$$P_c(D) = \sum_{k \in C} P(k) \prod_{i=1}^{n-1} P_c(k|W_i). \tag{2.42}$$

Z drugiej strony, prawdopodobieństwo poprawności klasyfikacji danego węzła W jest kombinacją liniową prawdopodobieństw poprawnej klasyfikacji przez badany węzeł obiektów dla poszczególnych klas:

$$P_c(W) = \frac{\sum_{k \in C(W)} P(k) P_c(k|W)}{\sum_{k \in C(W)} P(k)}, \tag{2.43}$$

gdzie $C(W)$ jest zbiorem etykiet klas przypisanych węzłowi W.

Optymalizacja dla danego węzła nie może więc być tym samym co optymalizacja dla całego drzewa, ponieważ to zależy nieliniowo od poszczególnych $P_c(k|W)$.

W przypadku ogólniejszej definicji drzewa (2.8) prawdopodobieństwa dla węzłów nie będących liśćmi mają jeszcze mniejszy związek z prawdopodobieństwem poprawności klasyfikacji drzewa, bo dotyczą tylko klasy dominującej w węźle ignorując pozostałe klasy, które mogą mieć znaczny wpływ na klasyfikację całego drzewa.

Nie ma zatem jednoznacznego przełożenia optymalizacji na poziomie węzła na optymalizację całego drzewa. Sprawia to, że rozwiązanie problemu optymalizacyjnego dla drzew wymaga użycia wyczerpującego szukania w przestrzeni drzew, co w praktyce jest niemożliwe ze względu na bardzo dużą złożoność obliczeniową i dlatego, w realnych zastosowaniach, stosuje się różne metody szukania heurystycznego. Problemy z przekładaniem się poprawności klasyfikacji węzłów na poprawność drzewa są przyczyną, że większość systemów budowania drzew klasyfikacji używa całkiem innych kryteriów oceny podziału węzłów na podwęzły niż po prostu jakość klasyfikacji (najczęściej są to kryteria wywodzące się z teorii informacji).

Czystość węzłów. Najczęściej stosowane kryteria oceniające jakość podziału wyliczają pewne miary *czystości* (*jednorodności*) węzłów, a właściwie ich *nieczystości* (*niejednorodności*). Dla węzła W, w który wpadają wektory należące do tej samej klasy, wartość takiego kryterium $I(W)$ jest równa zeru. Miary te przyjmują maksymalne wartości dla węzłów zawierających równoliczne klasy.

Dla danego kryterium I za najkorzystniejszy podział węzła W uznawany jest ten podział S, który maksymalizuje wartość przyrostu czystości dodawanych węzłów w stosunku do W, czyli

$$\Delta I(S,W) = I(W) - \sum_i p_i I(W_i^S) \tag{2.44}$$

gdzie W_i^S są podwęzłami węzła W wynikającymi z podziału S oraz p_i jest stosunkiem liczby wektorów zbioru treningowego wpadających do węzła W_i do liczby wektorów wpadających do węzła W.

Binaryzacja. Systemy budowania drzew binarnych, aby pozbyć się problemu niemożności uwzględniania mniej licznych klas w problemach wieloklasowych, stosują technikę podobną do tej, którą de Sa przyjął już w definicji drzew decyzji. Technika ta to *binaryzacja* (ang. *twoing*) , która polega na tym, że kryterium oceniające podział na klasy używa w wieloklasowych problemach nie oryginalnego zestawu klas, a dwóch meta-klas wynikających z połączenia różnych klas w dwie grupy tak, że każda z klas przypisana jest do grupy odpowiadającej podwęzłowi, do którego wpada większość wektorów z tej klasy.

Generalizacja. Ponieważ dla danego zbioru treningowego maksymalne drzewa (tzn. zbudowane tak, by uzyskać dla zbioru treningowego maksymalną poprawność klasyfikacji) prawie zawsze są nadmiernie dopasowane do danych treningowych, stosowane są różne metody mające na celu uzyskanie drzewa o zdolnościach generalizacyjnych.

Jedną z możliwości zapobiegania nadmiernej szczegółowości drzew jest stosowanie tzw. *kryteriów stopu*, czyli funkcji oceniających, czy dany węzeł należy dzielić czy nie. Są to rozwiązania bardzo szybkie, bo nie wymagają zbudowania nawet jednego pełnego drzewa, jednak w związku z trudnością globalnego określenia skutecznych kryteriów, trudno jest w ten sposób osiągnąć stawiany cel.

Jednym z najprostszych kryteriów stopu jest określenie minimalnej liczby wektorów, które mogą stanowić liść. Liczba ta może się jednak znacząco różnić dla różnych zbiorów danych. Co więcej, charakter tych samych danych może być różny w różnych miejscach przestrzeni i zbudowanie optymalnego drzewa może wymagać liści o różnych licznościach.

Bardziej skutecznym środkiem może być dodawanie do oceny nieczystości drzewa członu zależnego od rozmiaru drzewa (liczby węzłów). Jest to rozwiązanie zgodne z zasadą MDL. Wymaga ono jednak dodatkowego parametru kontrolującego na ile czyste ma być końcowe drzewo, a na ile mało złożone.

Inna metodą jest wykonanie testu (np. χ^2) dla sprawdzenia istotności poprawy czystości drzewa.

Wszystkie te kryteria mają skłonność do decydowania o zakończeniu rozwijania drzewa w sytuacjach, w których dalsze podziały mogłyby być źródłem istotnych informacji. Z tego powodu lepszym rozwiązaniem wydaje się być budowanie maksymalnego drzewa dla danego zbioru treningowego i *obcięcie* (ang. *pruning*) pewnych jego liści tak, by uzyskać dobrze generalizujące drzewo.

Ocena kształtu drzewa dającego największe prawdopodobieństwo uogólniania problemu jest zwykle wykonywana poprzez *uczenie kroswalidacyjne*. Polega ono na stosowaniu testu kroswalidacji wewnątrz zbioru treningowego tak, by w każdym przebiegu oceniać złożoność modelu dającą optymalne wyniki na danych nie używanych do uczenia. Do wewnętrznego testu można również użyć kroswa-

lidacji Monte Carlo, baggingu (p. rozdział 2.13.2) itp.

Parametryczność. Wiele algorytmów drzew decyzji to algorytmy *nieparametryczne* tzn. nie bazujące na żadnych założeniach dotyczących rozkładu danych. Podobnie jak metody minimalnoodległościowe, tak i drzewa, które dzielą przestrzeń na rozłączne zbiory nie wymagają żadnych założeń co do rozkładu danych, jeśli tylko nie korzystają pośrednio z metod parametrycznych (np. pewnych testów statystycznych).

2.9.1 CART

Classification and Regression Trees (CART), to jedna z najpopularniejszych i najskuteczniejszych metod drzew decyzji [23, 130, 30].

Jest to nieparametryczny algorytm, który tworzy drzewa binarne wykorzystując w podziałach zarówno cechy ciągłe jak i dyskretne. Dla cech ciągłych rozpatrywane są wszystkie możliwe podziały na dwa zbiory $(-\infty, a]$ oraz (a, ∞), a dla cech dyskretnych wszystkie możliwe podziały zbioru wartości cechy na dwa rozłączne i uzupełniające się podzbiory.

Każdemu węzłowi przypisuje się etykietę klasy dominującej w węźle, bądź wynikającą z oceny kosztów pomyłek.

Jako kryterium oceniające jakość podziałów CART stosuje przyrost czystości węzłów. Sugerowaną przez autorów miarą nieczystości jest tzw. *Gini index*:

$$I_G(W) = 1 - \sum_{c \in C} P(c|W)^2. \tag{2.45}$$

Możliwe jest także użycie miary entropii:

$$I_E(W) = -\sum_{c \in C} P(c|W) \log_2 P(c|W). \tag{2.46}$$

Niezależnie od zastosowanej miary nieczystości można stosować technikę binaryzacji. CART dokonuje wówczas podziału danych na dwie superklasy (grupy oryginalnych klas), by dla nich oceniać jakości podziałów. System usiłuje tak połączyć klasy w grupy, by powstałe superklasy były w miarę możliwości równoliczne.

W przypadku niepełności danych, dane dla których brakuje wartości wykorzystywanej w danym węźle są analizowane poprzez tzw. *podziały zastępcze* (ang. *surrogate splits*). Dla dokonanego podziału węzła wyznacza się dodatkowo podziały używające innych cech, które są maksymalnie zbliżone do wybranego i one (w odpowiedniej kolejności) są wykorzystywane w przypadku braku wartości cechy używanej w głównym warunku podziału.

System CART może wykorzystywać kryterium stopu (z zadaną minimalną liczbą wektorów w węźle), choć sugerowanym rozwiązaniem jest budowanie maksymalnego drzewa i ocena generalizacji w procesie kroswalidacji. W wewnętrznym teście wyznacza się optymalną wartość λ minimalizującą $\mathsf{Err}(D) + \lambda l_D$, gdzie l_D jest liczbą liści drzewa D. CART stosuje dwa warianty tej metody: 0-SE oraz 1-SE. Pierwszy z nich wybiera rzeczywiście drzewo o minimalnej wartości powyższego wyrażenia, drugi obcina drzewo maksymalnie pozwalając na większą wartość wyrażenia, ale nie większą od minimalnej o więcej niż wartość standardowego odchylenia wyznaczonego w procesie kroswalidacji.

W systemie CART możliwe jest również budowanie drzew dla problemów regresji.

2.9.2 ID3

Algorytm ID3 [150, 131] został oparty na teorii informacji. Jego poważną wadą, która bardzo ogranicza zastosowanie metody w praktyce, jest wymóg dyskretności wszystkich cech przestrzeni klasyfikacji. Jeśli chcemy użyć metody do problemów zdefiniowanych w przestrzeniach o cechach ciągłych, to trzeba najpierw zdyskretyzować wszystkie cechy. Etap dyskretyzacji jest wówczas etapem kluczowym dla ostatecznej klasyfikacji, więc ta sama metoda ID3 może dawać bardzo różne rezultaty dla różnych metod dyskretyzacji.

Kryterium oceniającym podziały jest kryterium przyrostu czystości (2.44). Miarą niejednorodności węzła jest miara entropii 2.46. Przyrost czystości jest nazywany wówczas *zyskiem informacyjnym* (ang. *information gain*).

Metoda polega na rekurencyjnym dzieleniu węzłów na podwęzły aż do uzyskania maksymalnego drzewa. W każdym kroku metoda dzieli dany węzeł na tyle podwęzłów ile wartości ma najbardziej informatywna cecha (cecha oferująca maksymalną redukcję entropii). Niekorzystną konsekwencją takiej strategii jest tendencja do częstszego wykorzystywania cech, które mają dużą (w stosunku do innych) liczbę możliwych wartości.

2.9.3 C4.5

System budowania drzew decyzji o nazwie C4.5, jest obok systemu CART jednym z dwóch najpopularniejszych i najczęściej stosowanych. Quinlan stworzył go [151] na bazie metody ID3. W istotnej części algorytmy te są identyczne. Różnice wprowadzone w C4.5 to:

- modyfikacja miary nieczystości węzłów,
- możliwość wykorzystania ciągłych atrybutów,

- metoda oczyszczania drzewa,
- możliwość klasyfikacji niepełnych danych.

Modyfikacja miary niejednorodności węzłów ma na celu uniknięcie niepożądanego efektu preferowania atrybutów o dużej liczbie możliwych wartości. Zamiast zysku informacyjnego (2.44) stosuje się *względny zysk* (ang. *gain ratio*) określony wzorem

$$\Delta I'(S,W) = \frac{\Delta I_E}{SI(S,W)}, \tag{2.47}$$

gdzie

$$SI(S,W) = -\sum_i p_i \log_2 p_i. \tag{2.48}$$

Tak jak w równaniu 2.44, p_i określa stosunek liczby wektorów wpadających do i-tego podwęzła do liczby wektorów w węźle W, a zatem $SI(S,W)$ (ang. *Split Information*) jest entropią rozkładu wynikającego z podziału S.

Dla atrybutów ciągłych algorytm C4.5 rozpatruje podobnie jak CART wszystkie możliwe podziały na dwa podzbiory zdeterminowane punktem podziału a. W przeciwieństwie do atrybutów dyskretnych, ciągłe mogą pojawiać się na wielu poziomach tej samej gałęzi drzewa. Dla każdego z możliwych podziałów ocenia się jego jakość mierząc wartość względnego zysku informacyjnego i wybiera ten, który maksymalizuje zysk.

Oczyszczanie (przycinanie) drzewa stosowane w C4.5 opiera się na statystycznej ocenie istotności różnicy błędu klasyfikacji dla danego węzła i jego podwęzłów. Zakładając dwumianowy rozkład liczby błędów szacuje się prawdopodobieństwo zmniejszenia błędu i obcina podziały, dla których to prawdopodobieństwo nie przekracza zadanego progu, ewentualnie zamienia poddrzewo o korzeniu w danym węźle jego najlepszym poddrzewem.

C4.5 potrafi operować na danych, w których brakuje pewnych wartości. W przypadku budowania drzewa ocena zysku nie uwzględnia danych, dla których brakuje wartości badanej cechy, a wyliczony zysk skaluje się mnożąc przez częstość występowania wartości tej cechy w próbie treningowej. Podział danych na podwęzły wprowadza wówczas wagi dla wektorów treningowych, które dla wektora z brakującą wartością atrybutu decyzyjnego odpowiadają rozkładowi pozostałych danych w podwęzłach. Stosownej modyfikacji podlegają wówczas współczynniki p_i ze wzoru 2.47 – zamiast mocy zbiorów liczy się sumy wag elementów tych zbiorów. Współczynniki p_i są uwzględniane również przy podejmowaniu decyzji na podstawie drzewa, by wyliczyć prawdopodobieństwa wpadania do poszczególnych węzłów oraz przynależenia do poszczególnych klas.

System C4.5 oprócz metody indukcji drzewa decyzji oferuje klasyfikator będący zbiorem reguł logiki klasycznej. Reguły traktowane są tutaj jako różny od

drzewa model klasyfikacji, ponieważ nie są one wierną reprezentacją drzewa. Reguły wiernie opisujące drzewo są poddawane procesowi oczyszczania: w każdej regule usuwane są przesłanki, których pominięcie nie powoduje spadku jakości klasyfikacji zbioru treningowego. Ponieważ oczyszczanie wykonywane jest dla każdej reguły z osobna, można w ten sposób otrzymać klasyfikator istotnie różny od drzewa (zwykle dający w testach niższe wartości poprawności).

Zmodyfikowana wersja algorytmu C4.5 nazywana *C5.0* bądź *See5* jest produktem komercyjnym, którego popularność jest zdecydowanie mniejsza niż C4.5, w związku z czym jej rezultaty nie są tak powszechnie dostępne jak rezultaty zastosowań jej poprzednika.

2.9.4 FACT i QUEST

FACT (*Fast Algorithm for Classification Trees*) [121] oraz QUEST (*Quick, Unbiased, Efficient, Statistical Tree*) [120] to drzewa decyzji bazujące na metodach statystycznych.

Są to metody parametryczne – wykorzystywane statystyki bazują na pewnych założeniach co do rozkładów danych.

Obydwa algorytmy dokonują podziałów dla cech ciągłych. Cechy dyskretne są konwertowane do ciągłych specjalnymi metodami. Wykonują one najpierw projekcję danego wymiaru do przestrzeni o wymiarowości równej liczbie symboli przetwarzanej cechy (symbolom odpowiadają stosowne wektory składające się z wielu zer i jednej jedynki), a następnie rzutowanie na odpowiednio wybraną prostą. Metoda QUEST stosuje tutaj nieco rozbudowaną w stosunku do FACT strategię.

Wybór cechy podziału jest w tych algorytmach wyraźnie oddzielony od samego podziału. W pierwszym etapie wybierana jest cecha podziału. FACT wybiera ją na podstawie analizy każdej cechy (dyskretnych po przekonwertowaniu do ciągłych) statystyką F znaną z metody analizy wariancji (ANOVA) [171, 18] – wybierana jest ta cecha, dla której wartość statystyki F jest maksymalna. QUEST również używa statystyki F do oceny cech ciągłych, jednak ocena cech dyskretnych wykonywana jest poprzez test χ^2 niezależności klasy i danej cechy. Celem jest uniknięcie faworyzowania zmiennych dyskretnych przy wyborze cechy podziału.

Dla danej cechy FACT dokonuje podziału na podstawie wyników liniowej dyskryminacji (LDA). Węzeł dzielony jest na tyle podwęzłów, ile klas jest w nim reprezentowanych. QUEST jest drzewem binarnym – przed dokonaniem podziału łączy klasy w dwie grupy wyznaczone metodą klasteryzacji *two-means* zastosowaną do zbioru średnich wyliczonych dla poszczególnych klas (jeśli średnie są jednakowe, to wyznacza się grupy tak, że klasa dominująca stanowi jedną z nich, a reszta klas drugą). Dodatkowo zamiast LDA, do wyznaczania podziału QUEST

wykorzystuje dyskryminację kwadratową (QDA). W wyniku QDA otrzymuje się dwa potencjalne punkty podziału, z których wybiera się ten, który jest bliższy średniej wartości jaką przyjmuje analizowana cecha w populacji wektorów należących do jednego z klastrów.

W obydwu algorytmach, kontrola złożoności drzew może się odbywać z wykorzystaniem kryterium stopu. QUEST potrafi także stosować w tym celu kroswalidację na wzór systemu CART.

2.9.5 Cal5

System indukcji drzew decyzji Cal5 [133, 134] przeznaczony jest dla przestrzeni cech ciągłych. Podstawowym elementem tej metody jest algorytm podziału cech ciągłych na przedziały wykorzystujący metody statystyczne do oceny jednorodności węzłów.

Istnieje jednak możliwość odpowiedniej adaptacji metody na użytek przestrzeni z cechami symbolicznymi – należy wówczas skorzystać z tych samych mechanizmów, które w przypadku cech ciągłych (w połączonym procesie dyskretyzacji i oceny przedziałów z niej wynikających) odpowiadają za ocenę wyznaczanych zbiorów i podejmowanie decyzji o zakończeniu (bądź nie) gałęzi drzewa.

Każdy węzeł drzewa Cal5 może mieć inną liczbę podwęzłów – jest ona wyznaczana automatycznie w procesie podziału zakresu danej cechy na przedziały i łączenia powstałych przedziałów.

Sposób działania metody jest uzależniony od dwóch parametrów: progu decyzyjnego S (określającego minimalne prawdopodobieństwo poprawnej klasyfikacji danego węzła, które pozwala uznać go za liść) oraz poziomu α dla statystycznych testów wykorzystywanych w procesie dyskretyzacji.

Drzewo budowane jest rekurencyjnie poprzez dokonywanie podziałów węzłów na podwęzły. Analiza każdego węzła, który nie jest liściem składa się z trzech kroków:

1. wyboru najlepszego atrybutu,
2. dyskretyzacji,
3. łączenia przedziałów wynikających z dyskretyzacji.

Wybór najlepszego atrybutu. Decyzja o wyborze cechy, której dotyczyć będą warunki dzielące węzeł może być podejmowana w Cal5 na bazie oceny statystycznej bądź opartej na entropii.

Metoda statystyczna polega na ocenie każdej z cech ilorazem

$$Q = \frac{A^2}{A^2 + D^2}, \tag{2.49}$$

gdzie D jest średnią kwadratów wariancji oszacowanej na podstawie wektorów treningowych poszczególnych klas, oraz A jest średnią kwadratów odległości pomiędzy średnimi wartościami cechy dla różnych klas.

Metoda oparta na entropii ocenia cechy wyliczając średnią entropię podwęzłów ważoną proporcjonalnie do liczności przedziałów, czyli odpowiednią entropię warunkową, a zatem jest to ta sama metoda, która jest używana np. w drzewie ID3, czyli uwzględniająca zysk informacyjny. Zastosowanie tej metody wymaga uprzedniej dyskretyzacji ocenianych cech, a zatem niejako odwraca kolejność tych dwóch etapów.

Dyskretyzacja. Pierwszym krokiem do dyskretyzacji cech jest posortowanie wektorów treningowych według rosnących wartości badanej cechy. Następnie analizowane są przedziały odpowiednio rozszerzane, by obejmowały kolejne (od lewej do prawej) wektory. Analiza przedziału x polega na sprawdzaniu dwóch hipotez:

H1 – istnieje klasa c, dla której $p(c|x) \geqslant S$,

H2 – dla wszystkich klas $c \in C$, $p(c|x) < S$.

Weryfikacja hipotez odbywa się poprzez wyznaczenie przedziału ufności dla zadanego poziomu α (z wykorzystaniem nierówności Czebyszewa, oraz przy założeniu rozkładu Bernoulliego dla każdej z klas) i sprawdzenie, czy przedział ten w całości leży po odpowiedniej stronie progu S. Może zajść jedna z trzech sytuacji:

1. Hipoteza H1 jest prawdziwa. Wówczas przedział jest uznawany za wyznaczony – będzie wyznaczał liść drzewa decyzji o etykiecie klasy, dla której spełniony jest warunek hipotezy. Wyznaczane są następne przedziały dla kolejnych danych w uporządkowanej liście.

2. Hipoteza H2 jest prawdziwa. Wówczas przedział jest również uznawany za wyznaczony, ale będzie wyznaczał węzeł przeznaczony do dalszego podziału, bo żadna z klas nie dominuje w wystarczający sposób.

3. Ani H1, ani H2 nie są prawdziwe. Rozszerzamy przedział o następny przypadek z uporządkowanej listy. Jeśli na liście nie ma więcej wektorów, to dany przedział wyznacza liść o etykiecie klasy dominującej.

Łączenie przedziałów. Po dyskretyzacji sąsiednie przedziały są łączone jeśli oba są liśćmi o tej samej klasie dominującej. Również kiedy przedziały nie są liśćmi (są przeznaczone do dalszego podziału), można je połączyć jeśli zawierają

ten sam zestaw klas (po eliminacji tych istotnie mniej reprezentowanych niż w przypadku równego podziału – eliminacja ta opiera się również na wyniku testu statystycznego).

2.9.6 Inne algorytmy drzew decyzji

Istnieje również wiele innych (niż opisane wyżej) algorytmów indukcji drzew decyzji proponowanych przez różnych autorów.

Część z nich, to różne modyfikacje podstawowych algorytmów takich jak ID3. Na przykład system NewID stosuje podobnie jak ID3 miarę zysku informacyjnego do wyboru cech używanych do kolejnych podziałów, przy czym pozwala na używanie także ciągłych atrybutów (technika jest taka sama jak w C4.5).

TDDT (*Top-Down Decision Trees*) to algorytm generowania drzew decyzji dostępny w bibliotece MLC++ [113]. Jest to metoda bardzo podobna do C4.5 – najistotniejszą różnicą jest używanie miary zysku informacyjnego podzielonego przez logarytm z liczby generowanych podziałem podwęzłów drzewa (taka modyfikacja ma na celu unikanie nadmiernego „rozdrabniania" węzłów poprzez preferowanie cech, które oferują duży zysk informacyjny jako skutek dużej liczności zbioru możliwych symboli).

W literaturze spotkać można także szereg algorytmów indukcji drzew decyzji, w których funkcje przynależności do węzłów wykorzystują więcej niż jedną cechę. Takie systemy oferować mogą atrakcyjne wyniki klasyfikacji, choć ich interpretacja jest na ogół zdecydowanie trudniejsza niż w przypadku drzew o granicach decyzji prostopadłych do osi przestrzeni klasyfikacji.

Jednym z takich rozwiązań są drzewa wykorzystujące w funkcjach przynależności węzłów kombinacje liniowe znajdowane z użyciem kryteriów dipolowych [16], które w swoich założeniach są bliskie prezentowanemu w rozdziale 3 kryterium separowalności.

Innym podejściem wykorzystującym liniowe kombinacje cech są Drzewa Maszyn Liniowych (ang. *Linear Machine Decision Trees* – LMDT) [180, 24, 25]. Wykorzystują one w każdym węźle drzewa tzw. *maszyny liniowe*, czyli metody wyznaczania liniowych funkcji dyskryminujących (po jednej dla każdej z klas). Dodatkowo zabiegają o prostotę opisu poprzez eliminację zmiennych – kiedy liniowa funkcja dyskryminująca jest bliska finalnej, rozwiązanie jest zapamiętywane, a następnie usuwana jest najmniej istotna cecha i szukanie atrakcyjnej funkcji dyskryminującej jest kontynuowane w tak zredukowanej przestrzeni (jeśli bez powodzenia, to powraca się do zapamiętanego rozwiązania).

Kolejnym ze znanych rozwiązań są „ukośne" drzewa OC1 (ang. *Oblique Classifier*) [136, 135]. System ten szuka drzew metodą wspinaczki (ang. *hill climbing*), a kombinacje liniowe wykorzystywane w węzłach wyznacza procesem szukania

łączącym metody heurystyczne i niedeterministyczne (dla wychodzenia z minimów lokalnych).

Drzewami decyzji o bardziej złożonej postaci są również *drzewa z opcjami* (ang. *Option decision trees*) [28, 112]. Pozwalają one na wyprowadzanie z tego samego węzła kilku alternatywnych gałęzi, by potem decydować o klasyfikacji przez zastosowanie odpowiedniego rachunku prawdopodobieństw. Jest to metoda na reprezentację wielu drzew w jednym – reprezentacja jest prostsza niż dla wielu osobnych drzew, bowiem drzewa mogą zawierać wspólne fragmenty.

2.10 Metody indukcji reguł

Reguły klasyfikacji mogą być generowane bezpośrednio z danych (nie poprzez drzewa decyzji).

Bardzo skutecznym dla wielu danych jest system PVM (Predictive Value Maximization), który opracowali Weiss i Kapouleas [182]. Ponieważ system ten wykonuje pełne przeszukiwanie przestrzeni rozwiązań, bardzo dobrze radzi sobie z małymi zbiorami danych, jednak użycie go dla większych zadań może stwarzać problemy kombinatoryczne.

Jak bardzo skuteczne potrafią być pojedyncze proste reguły, dowodzi algorytm 1R, który zaproponował Holte [90]. Jest to bardzo prosta metoda polegająca na tworzeniu reguł klasyfikacji używających tylko jednego atrybutu. Pozwala ona odkrywać proste zależności pomiędzy cechami a klasami, choć w testach generalizacji zwykle systemy bardziej złożone niż 1R są istotnie dokładniejsze.

Zadanie generowania regułowych opisów danych jest również podejmowane poprzez definiowanie bardzo ogólnych i złożonych systemów logicznych, które dysponują przeróżnymi formami opisu. Jeden z takich systemów został zaproponowany w [17]. Jest to ogólny schemat konstruowania predykatów, którymi można wyrażać najróżniejsze własności obiektów o skomplikowanej strukturze (wykraczającej poza definicję 2.1 przestrzeni klasyfikacji). Niestety bogaty język wiąże się z dużą złożonością algorytmów szukania trafnych opisów. Są to więc systemy o teoretycznie bardzo dużych możliwościach, ale w praktyce ze względu na bardzo dużą złożoność obliczeniową, mogą być efektywnie stosowane tylko dla małych zadań.

Kolejne podrozdziały opisują trzy różne metody indukcji reguł z danych (AQ, CN2 oraz ITRULE), które należą do najpopularniejszych algorytmów tej grupy.

2.10.1 AQ

System AQ [128, 185] rozwijany od 1969 roku doczekał się wielu różnych wersji algorytmu, jednak ogólny schemat działania systemu pozostaje bez większych

zmian.

Język używany przez autorów do opisu systemu posługuje się pojęciami takimi jak *selektory*, *kompleksy* czy *gwiazdy*, które są kluczowe dla rozumienia algorytmu. Bez wprowadzania specjalnego formalizmu można określić selektory jako alternatywy warunków dotyczących tej samej cechy, kompleksy jako koniunkcje selektorów (dodatkowo kompleks pusty jest warunkiem, który jest spełniony przez każdy wektor), natomiast gwiazdy jako zbiory kompleksów. Ponieważ selektory i kompleksy są pewnymi warunkami logicznymi, więc można mówić o ich spełnianiu przez wektory – częściej mówi się jednak równoważnie o pokrywaniu wektorów przez selektory i kompleksy.

Szukanie reguł polega na wyborze wektora (zwanego *ziarnem*) i budowaniu na jego podstawie alternatywnych reguł pokrywających jak najwięcej przypadków z danej klasy (stąd strategię tą nazywa się często metodą *rozrostu ziarna*).

AQ jest algorytmem szukającym zestawów reguł klasyfikacji danych zdefiniowanych w przestrzeni cech dyskretnych. W podstawowej wersji generuje reguły opisujące jedną z klas (wektory treningowe z tej klasy nazywa się przykładami *pozytywnymi*, a pozostałe *negatywnymi*). Dla problemów wieloklasowych wystarczy powtórzyć algorytm odpowiednią liczbę razy dla różnych klas. Schemat metody przedstawia algorytm 2.3.

Algorytm 2.3 (Indukcja reguł metodą AQ)

► ***Dane:*** *Przestrzeń cech dyskretnych X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$, szerokość wiązki $w \in N$.*

◄ ***Wynik:*** *Zestaw reguł klasyfikacji $\mathcal{R}$.*

1. $\mathcal{R} \leftarrow \emptyset$
2. *Tak długo jak istnieją przykłady pozytywne nie pokryte przez żadną z reguł w $\mathcal{R}$ powtarzamy:*
   (a) *Wybieramy ziarno spośród przykładów pozytywnych.*
   (b) *Generujemy (algorytmem 2.4 z szerokością wiązki w) gwiazdę pokrywającą wybrane ziarno i nie pokrywającą żadnego przykładu negatywnego.*
   (c) *Dodajemy do $\mathcal{R}$ regułę, której poprzednikiem jest najlepszy (według pewnych kryteriów) kompleks gwiazdy, a następnikiem stwierdzenie przynależności do analizowanej klasy.*
3. *Wynikiem algorytmu jest zbiór $\mathcal{R}$ reguł klasyfikacji.*

Kryteria używane do oceny kompleksów w punkcie 2c algorytmu 2.3 mogą być formułowane na różne sposoby – podstawowym jest porządek antyleksykograficzny par określających liczby przykładów pozytywnych pokrywanych przez kompleks oraz przykładów pokrywanych przez kompleks ale nie pokrywanych przez żadną z dotychczas wybranych reguł.

Te same kryteria służą do określania, które kompleksy gwiazdy zostaną zakwalifikowane do wiązki w procesie szukania przedstawionym algorytmem 2.4

Algorytm 2.4 (Szukanie gwiazdy dla metody AQ)

► ***Dane:*** *Przestrzeń cech dyskretnych X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$, szerokość wiązki $w \in N$, ziarno $z \in T$.*

◄ ***Wynik:*** *Gwiazda $\mathcal{G}$.*

1. *$\mathcal{G} \leftarrow$ {kompleks pusty}*
2. *Tak długo jak dowolny przykład negatywny jest pokryty przez dowolny kompleks gwiazdy $\mathcal{G}$ powtarzamy:*
   (a) *Wybieramy przykład negatywny n spełniający powyższy warunek.*
   (b) *Generujemy zbiór $\mathcal{M}$ maksymalnie ogólnych (pokrywających maksymalne obszary przestrzeni) kompleksów, które pokrywają z ale nie pokrywają n.*
   (c) *Tworzymy zbiór $\mathcal{P}$ kompleksów będących przecięciami kompleksu gwiazdy $\mathcal{G}$ oraz kompleksu ze zbioru $\mathcal{M}$.*
   (d) *$\mathcal{G} \leftarrow$ wiązka w najlepszych kompleksów z $\mathcal{P}$.*
3. *Wynikiem algorytmu jest gwiazda $\mathcal{G}$.*

2.10.2 CN2

Algorytm CN2 [33] został opracowany by połączyć zalety metody AQ oraz technik stosowanych w budowaniu drzew ID3. Jest to metoda indukcji reguł bardzo podobna do AQ, ale wprowadzająca szereg zmian (uogólnień, z których najważniejsze są następujące:

- Zrezygnowano z wymogu stuprocentowej znamienności reguł. W ten sposób algorytm jest mniej wrażliwy na zaszumienie danych.

- Generowanie kolejnej reguły uwzględnia tylko te wektory zbioru treningowego, które nie są pokrywane dotychczas wybranymi regułami. Dzięki temu reguły jednego zestawu mogą opisywać różne klasy. Taka technika sprawia, że istotną staje się kolejność reguł w zestawie (choć istnieje wersja z nieistotnym porządkiem reguł [32] – wówczas sposób ich generowania jest taki sam jak w AQ).

- Funkcja oceniająca jakość kompleksów liczy dla nich entropię rozkładu danych (w ten sposób można akceptować kompleksy o niższej znamienności niż 100%).

- W procesie szukania gwiazdy rozważa się wszystkie możliwe specjalizacje kompleksów, które polegają na dodaniu warunku do koniunkcji lub usunięciu warunku z jednego z selektorów. Jest to sposób szukania, który znacznie zwiększa złożoność obliczeniową w stosunku do metody AQ.

- Na wzór metod drzew decyzji stosuje się oczyszczanie (*pruning*) reguł na podstawie testu statystycznej istotności opartego na statystyce entropii:

$$2 \cdot \sum_{i=1}^{k} f_i \log \frac{f_i}{e_i}, \tag{2.50}$$

  gdzie $f_1, \ldots, f_k$ to rozkład częstości występowania przypadków z poszczególnych klas w zbiorze wektorów treningowych pokrywanych kompleksem, a $e_1, \ldots, e_k$ to oczekiwane częstości dla równie licznej próbki o takim samym rozkładzie klas jak w całym zbiorze treningowym.

- Dla przypadków nie pokrytych żadną regułą wprowadza się warunek „w przeciwnym przypadku" przypisujący wektor do klasy dominującej w zbiorze treningowym.

- Zmodyfikowana wersja algorytmu [32] korzysta z oszacowania Laplace'a (p. rozdział 3.10) prawdopodobieństw używanych w mierze entropii, co pozwala uzyskać istotnie lepsze wyniki.

Mimo stosowania metod oczyszczania zbioru reguł algorytm CN2 podobnie jak AQ na tendencję do nadmiernego dopasowywania się do danych treningowych, przez co zdarza się, że uzyskuje w testach generalizacji gorsze wyniki [32] niż klasyfikator podstawowy.

2.10.3 ITRULE

ITRULE (*Information-Theoretic Rule Induction*) [72, 74, 73, 166] jest algorytmem indukcji reguł, który bazuje na teorii informacji. Kryteriami teorii informacji

ocenia reguły postaci:

$$Y_1 = y_1 \wedge \ldots \wedge Y_n = y_n \rightarrow X = x \text{ z prawdopodobieństwem } p. \tag{2.51}$$

Ponieważ system operuje w przestrzeni cech dyskretnych, to można generować reguły, które w następniku mają własności dotyczące którejkolwiek z cech (każdą z cech możemy traktować jako zbiór klas), choć w systemach klasyfikacji szukanie będzie ograniczone do tych reguł, które wnioskują na temat klas obiektów.

Reguły oceniane są tzw. miarą J, której wiele szczegółowych własności przedstawiono w [167]. Bez utraty ogólności można rozpatrywać tę miarę i algorytm dla reguł z jedną przesłanką $Y = y$. Wówczas miara J dla takiej reguły określona jest wzorem

$$J(X;Y=y) \stackrel{\text{def}}{=} P(Y=y)\sum_x P(X=x|Y=y)\log\frac{P(X=x|Y=y)}{P(X=x)}. \tag{2.52}$$

Suma wartości tej miary, dla wszystkich możliwych wartości y zmiennej Y, jest równa *informacji wzajemnej* pomiędzy zmiennymi X oraz Y, która jest zdefiniowana jako różnica stosownych entropii:

$$I(X;Y) \stackrel{\text{def}}{=} H(X) - H(X|Y). \tag{2.53}$$

Algorytm ITRULE polega na przeszukiwaniu przestrzeni reguł w celu znalezienia zadanej liczby reguł o stosunkowo najwyższych wartościach miary J. Do tego celu używana jest metoda szukania w głąb, która jest wzbogacona o pewne kryteria mające na celu uproszczenie procesu.

W danym problemie klasyfikacji maksymalna liczba reguł do oszacowania wynosi $m((2m+1)^{n-1}-1)$, gdzie n jest liczbą cech, a m liczbą wartości każdej z cech (dla uproszczenia zakłada się jednakową dla wszystkich cech). W praktyce przeszukanie całej przestrzeni nie jest możliwe, więc konieczne są pewne ograniczenia.

Korzystając z własności miary J (udowodnionych na bazie teorii informacji) można definiować ograniczenia, które mogą stwierdzać, czy dalsza specyfikacja danej reguły może przynieść pożądany wzrost miary J, a co za tym idzie, czy należy proces szukania kontynuować w głąb, czy też można daną gałąź drzewa poszukiwań pominąć. Innym z możliwych ograniczeń może być również zadana maksymalna liczba przesłanek reguły.

Do szacowania prawdopodobieństw, które definiują miarę J używa się metody m-oszacowania (p. rozdział 3.10).

2.11 Systemy logiki rozmytej

Logika rozmyta jest terminem bardzo szerokim, bo obejmuje wiele różnych zagadnień, w których rozpatruje się możliwości wnioskowania na podstawie nie-

jednoznacznych i nieprecyzyjnych pojęć, albo używa się wartości logicznych z przedziału [0,1], które są interpretowane jako stopień prawdziwości stwierdzeń. Można więc patrzeć na logikę rozmytą jak na teorię uzupełniającą rachunek prawdopodobieństwa w opisywaniu nieprecyzyjnych pojęć.

U podstaw rozumowania rozmytego stoi teoria zbiorów rozmytych [186, 100], która wprowadza pojęcie stopnia przynależności do zbioru i definiuje zbiory poprzez funkcje przynależności o przeciwdziedzinie będącej przedziałem [0,1]. Teoria ta znalazła szerokie zastosowanie do podejmowania decyzji w warunkach niepewności [7]. Udowodniono [115], że systemy rozmyte są uniwersalnymi aproksymatorami, więc mogą być bardzo skuteczne w rozwiązywaniu problemów aproksymacji i klasyfikacji.

Najszersze zastosowanie teoria zbiorów rozmytych znalazła jednak w systemach sterowania. Wiele jest różnych podejść do zadania sterowania w warunkach rozmytości. Większość z nich polega na wykorzystaniu wiedzy niezbędnej do sterowania procesem by tworzyć sterowniki z powodzeniem wykonujące to zadanie, mimo nieznajomości modelu sterowanego procesu (jako funkcji wiążącej wejście z wyjściem). Baza wiedzy będąca podstawą sterowania definiuje wówczas wszelkie funkcje przynależności oraz rozmyte reguły, na podstawie których odbywa się sterowanie.

Podejście takie nazywa się *deskryptywnym* w przeciwieństwie do *preskryptywnego* [101], w którym zadaniem jest wyznaczanie optymalnych metod sterowania, gdy znane są: model procesu określający wyjście jako funkcję wejścia oraz funkcja celu.

Jednak także przy podejściu deskryptywnym można spotkać nietrywialne problemy. Na przykład, w pewnych sytuacjach umiejętność sterowania wynika z doświadczenia operatora i niełatwo jest opisać ją w postaci reguł. Wówczas można określać reguły sterujące w procesach adaptacji na podstawie dostępnych danych empirycznych, a więc problem ten jest w zasadzie równoważny problemowi klasyfikacji.

Wiele metod wyznaczania reguł rozmytych bazuje na sieciach neuronowych. Stosunkowo łatwo jest opisać takimi regułami działanie sieci neuronowej typu MLP. Niestety, zwykle liczba otrzymanych w ten sposób reguł jest duża, a więc ich zrozumienie dość trudne.

Sieci neuronowe oparte o separowalne zlokalizowane funkcje transferu są równoważne systemom logiki rozmytej [96] jako, że funkcja transferu każdego węzła może być wprost zapisana w języku logiki rozmytej. Ogólną propozycję systemu neurorozmytego opartego na separowalnych funkcjach przedstawiono w pracach [41, 50]. Ogólną dyskusję na temat indukcji reguł przy użyciu zlokalizowanych funkcji transferu przeprowadzili w swojej pracy [5] Andrews i Geva. Takie systemy neurorozmyte powinny mieć zdecydowaną przewagę w zastosowaniach do indukcji reguł, ponieważ używają bogatszej reprezentacji (reguły logi-

ki klasycznej są podzbiorem reguł rozmytych). Znanych jest wiele takich metod [50, 137, 139, 82, 179], jednak w praktyce rzadko wykorzystuje się je do indukcji klasycznych reguł. Główną przyczyną są trudności ze znalezieniem optymalnego rozwiązania dla licznych parametrów adaptacyjnych [104, 105]. Funkcja błędu dla klasycznych reguł logicznych wydaje się mieć wiele minimów lokalnych, w których metody gradientowe łatwo grzęzną.

Znane są także metody tworzące reguły rozmyte bezpośrednio z danych. Do wyznaczania zmiennych lingwistycznych można zastosować na przykład metody klasteryzacji [31]. Można też wykorzystać metody dyskretyzacji (używane w systemach indukcji reguł logiki klasycznej) a następnie dokonać rozmycia wyznaczonych przedziałów [27]. Inne podejście stosuje technikę hierarchicznego generowania rozmytych zmiennych lingwistycznych [106]. Istnieją również systemy hybrydowe takie jak rozmyte drzewa decyzji generowane z użyciem algorytmów ewolucyjnych [97].

2.12 Systemy zbiorów przybliżonych

Teoria zbiorów przybliżonych [142, 143, 144, 114] została przedstawiona przez Pawlaka w 1982 roku. Jest jednym ze sposobów uwzględniania sprzeczności, które mogą pojawiać się w danych.

Punktem wyjścia dla rozważań jest tzw. *przestrzeń przybliżeń* zdefiniowana jako para $A = (U,R)$, gdzie U jest zbiorem zwanym *uniwersum*, a $R \subseteq U \times U$ jest relacją równoważności nazywaną tutaj *relacją nierozróżnialności*. Klasy abstrakcji $[x]_R$ relacji R dla $x \in U$ są nazywane *zbiorami elementarnymi* w A, a ich skończone sumy *zbiorami definiowalnymi* w A.

Dla każdego zbioru $X \subseteq U$ definiuje się jego dwa przybliżenia: *dolne przybliżenie* X w A to zbiór

$$\{x \in X : [x]_R \subseteq X\}, \tag{2.54}$$

natomiast *górne przybliżenie* X w A to zbiór

$$\{x \in X : [x]_R \cap X \neq \emptyset\}. \tag{2.55}$$

Innymi słowy dolne przybliżenie zbioru X, to maksymalny zbiór definiowalny, który się w nim zawiera, a górne, to minimalny zbiór definiowalny zawierający X.

Zbiorem przybliżonym w A jest każda rodzina podzbiorów zbioru U o tych samych dolnych i górnych przybliżeniach. Zbiór przybliżony, dla którego te dwa przybliżenia są równe nazywany jest *zbiorem dokładnym*.

Na podstawie teorii zbiorów przybliżonych stworzono wiele różnych systemów analizy danych, na przykład LERS (jego idea została przedstawiona poniżej), Rosetta [141] czy Grobian [58].

Systemy wykorzystujące tę teorię, prowadzą z natury do zbioru reguł, jednak zwykle potrzebują one dodatkowych procedur dyskretyzacji ciągłych atrybutów oraz dają dużą liczbę reguł.

LERS. Jednym z systemów tworzących reguły klasyfikacji z wykorzystaniem teorii zbiorów przybliżonych jest LERS (ang. *Learning from Examples based on Rough Sets*) [80, 81]. Podobnie jak i dla innych systemów opartych na tej teorii, naturalnymi dla tej metody przestrzeniami klasyfikacji są przestrzenie cech dyskretnych.

Działanie systemu LERS polega na szukaniu minimalnego opisu dla każdej z klas reprezentowanych w zbiorze treningowym.

Dla danej przestrzeni cech dyskretnych $X = X_1 \times \cdots \times X_n$, *parą atrybut–wartość* jest każda para $f = (i, v)$, taka, że $i \in \{1, \ldots, n\}$ oraz $v \in X_i$. Symbol $[f]$ oznacza zbiór wektorów przestrzeni X, dla których cecha i przyjmuje wartość v:

$$[f] \stackrel{\text{ozn}}{=} \{x \in X : x_i = v\} \tag{2.56}$$

Minimalnym kompleksem dla zbioru $T \in X$ nazywa się minimalny zbiór F par atrybut–wartość taki, że $[F] \neq \emptyset$ oraz $[F] \subseteq T$, gdzie

$$[F] \stackrel{\text{ozn}}{=} \bigcap_{f \in F} [f]. \tag{2.57}$$

Rodzina $\mathscr{F}$ niepustych zbiorów par atrybut–wartość nazywana jest *lokalnym pokryciem* zbioru $T \in X$ o ile spełnione są warunki:

1. $\underset{F \in \mathscr{F}}{\forall} F$ jest minimalnym kompleksem dla T.
2. $\bigcup_{F \in \mathscr{F}} [F] = T$.
3. $\mathscr{F}$ jest minimalny (w sensie $|\mathscr{F}|$).

Algorytm LEM2 (podstawowy algorytm systemu LERS), dla każdej z klas danego zbioru treningowego T, produkuje dwa zestawy reguł – *pewny* i *możliwy* (ang. *certain* i *possible*), odpowiednio dla dolnego i górnego przybliżenia zbioru T. Dla każdego z przybliżeń zbioru wektorów z T należących do danej klasy, metoda LEM2 szuka lokalnego pokrycia, co daje w efekcie zbiór reguł klasyfikacji.

Aby zastosować algorytm LEM2 w zadaniach klasyfikacji zdefiniowanych w przestrzeniach z cechami ciągłymi, można przed użyciem metody poddać cechy ciągłe dyskretyzacji. Atrakcyjniejszym rozwiązaniem jest jednak wyznaczanie podziałów dla cech ciągłych w trakcie działania algorytmu – taką strategię stosuje na przykład metoda MODLEM [168].

2.13 Homogeniczne i heterogeniczne systemy złożone

Rosnące możliwości sprzętu komputerowego pozwalają na stosowanie coraz bardziej złożonych metod również w problemach klasyfikacji. Pojedyncze modele ustępują coraz częściej miejsca systemom złożonym.

W przypadku niedeterministycznych systemów klasyfikacji możliwość wielokrotnego uruchomienia algorytmu tworzącego model jest bardzo cenna, bo pozwala na tworzenie wzajemnie alternatywnych modeli, które mogą być użyte jako komponenty większego systemu.

Także systemy, które są deterministyczne, ale niestabilne tzn. przy małych zmianach zbioru treningowego potrafią generować istotnie różniące się rozwiązania, mogą być użyte do wielokrotnego trenowania na różnych próbkach danych w celu uzyskania alternatywnych rozwiązań.

Techniki wielokrotnego używania i wybierania danych treningowych w celu uzyskania lepszych modeli klasyfikujących noszą w języku angielskim wspólną nazwę *arcing*[3] (*adaptive reweighting and combining*).

Już na bazie jednego systemu poprzez wielokrotne uczenie można uzyskać znaczny przyrost dokładności klasyfikacji. Złożone systemy heterogeniczne mają jeszcze większą szansę na uzyskiwanie dobrych wyników w bardzo różnych zadaniach. Należy jednak mieć na względzie, że systemy złożone znacznie utrudniają (jeśli nie całkowicie uniemożliwiają) uzyskiwanie zrozumiałych wyjaśnień podejmowanych decyzji.

Opisy różnych techniki służących tworzeniu skutecznych klasyfikatorów złożonych można znaleźć m.in. w [57, 37].

Różne metody z tej rodziny mogą też być bardzo przydatne w tworzeniu systemów uczących się na poziomie *meta* (ang. *meta learning*), czyli szukających parametrów modeli odpowiednich dla danego zadania. Jest to dzisiaj bardzo szybko rozwijająca się dziedzina, która ma szanse zrodzić wiele bardzo interesujących systemów.

2.13.1 Bootstrap

Bootstrap to technika polegająca na generowaniu na podstawie pewnego zbioru danych innych zbiorów o tej samej mocy poprzez losowanie oryginalnych wektorów z powtórzeniami. Technika ta, poprzez jej wielokrotne powtórzenie, może być wykorzystana do szacowania wartości różnych parametrów (również średniej dokładności systemu klasyfikacji – p. rozdział 2.2.3).

[3]Bardzo trudno znaleźć w literaturze polskie odpowiedniki przedstawianych tutaj terminów. Angielskie wyrażenia są najczęściej wstawiane wprost do zdań wyrażanych w języku polskim.

2.13.2 Bagging

Słowo *bagging* pochodzi od wyrażenia *bootstrap aggregation*. Metoda ta polega na wielokrotnym zastosowaniu techniki *bootstrap* do wygenerowania odpowiedniej liczby zbiorów treningowych i wielokrotnym uczeniu danego systemu po to by ostateczną decyzję podejmować na podstawie głosowania wszystkich stworzonych modeli. Metoda ta jest szczególnie skuteczna w przypadku systemów niestabilnych (do których należą również drzewa decyzji) [152, 19, 22].

2.13.3 Boosting

Strategia boostingu polega na wielokrotnym tworzeniu próbek treningowych w taki sposób, by każdy następny zbiór (a co za tym idzie i budowany model) uwzględniał przede wszystkim te obszary przestrzeni, w których dotychczas zbudowane modele nie działają dobrze.

Jedną z najpopularniejszych metod należących do tej grupy jest *AdaBoost* (od *Adaptive Boosting*) [65, 64]:

Algorytm 2.5 (AdaBoost)

► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, system klasyfikacji S, zbiór treningowy $T = \{(x_1, c_1), \ldots, (x_n, c_n)\} \subseteq X \times C$, liczba modeli składowych $k_{\max} \in N$.*

◄ ***Wynik:*** *Klasyfikator.*

1. $k \leftarrow 1$
2. $W_1(i) = \frac{1}{n}$ *dla* $i = 1, \ldots, n$
3. *Tak długo jak* $k \leqslant k_{\max}$ *powtarzamy:*
   - *Tworzymy zbiór T_k poprzez wylosowanie ze zbioru T n elementów zgodnie z rozkładem W_k*
   - *Trenujemy system S na danych T_k uzyskując model M_k*
   - $E_k \leftarrow \sum_{i=0}^{n} W_k(i) \cdot \mathbf{1}_{M_k(x_i) \neq c_i}$
   - *Jeżeli* $E_k > 0.5$*, to przerywamy wykonywanie pętli.*
   - $\beta_k \leftarrow \frac{E_k}{1-E_k}$
   - *Dla każdego* $i = 1, \ldots, n$

$$W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} \beta_k & \text{o ile } M_k(x_i) = c_i \\ 1 & \text{w przeciwnym przypadku,} \end{cases} \tag{2.58}$$

   gdzie Z_k jest stałą renormalizującą dobraną tak, aby $\sum_{i=0}^{n} W_{k+1}(i) = 1$.

- $k \leftarrow k+1$

4. *Wynikiem jest klasyfikator*

$$f : x \longmapsto \operatorname*{argmax}_{c \in C} \sum_{i=1}^{k-1} \log \frac{1}{\beta_k} \cdot \mathbf{1}_{M_k(x)=c}. \tag{2.59}$$

Różne warianty boostingu znalazły wiele skutecznych zastosowań do poprawiania dokładności klasyfikatorów (zwłaszcza drzew decyzji) [152, 20, 21].

2.13.4 Komitety

Klasyfikator złożony z kilku innych klasyfikatorów i podejmujący decyzje na podstawie głosowania swoich składowych nazywa się *komitetem*. Komitety, z którymi mamy do czynienia w przypadku baggingu czy boostingu są komitetami homogenicznymi. Komitety heterogeniczne, których członkami są klasyfikatory wygenerowane przez odległe od siebie ideowo systemy, zasługują na szczególne zainteresowanie, bo pozwalają eksplorować dany problem na wiele różnych sposobów jednocześnie, więc mogą być skutecznym narzędziem wydobywania wiedzy z danych.

W najprostszej wersji komitetu wyznaczanie rezultatu głosowania polega na zliczaniu głosów i wybieraniu klasy na którą zagłosowało najwięcej klasyfikatorów (a jeśli takiej nie ma to jednej z klas o maksymalnej liczbie głosów).

Bardziej naturalnym dla komitetów jest traktowanie ich jako klasyfikatory probabilistyczne – łatwo wyznaczyć prawdopodobieństwa przynależności do poszczególnych klas. Jeśli członkowie komitetu są klasyfikatorami probabilistycznymi, to również łatwo można wyliczyć prawdopodobieństwa wynikające z głosowania jako odpowiednie wartości średnie.

Można także próbować przypisywać poszczególnym członkom komitetu kompetencje – nie globalnie, jak to ma miejsce w boostingu, ale lokalnie, przez określanie obszarów kompetencji (czy też niekompetencji) dla każdego z klasyfikatorów. Na ogół wymaga to jednak dodatkowego procesu adaptacyjnego.

2.13.5 Stacking

Poprawienie rezultatów klasyfikacji można uzyskać także tworząc dla danej grupy nauczonych modeli dodatkowy klasyfikator, dla którego danymi treningowymi byłyby odpowiedzi pobrane od każdego z klasyfikatorów z tej grupy. Tworzymy w ten sposób pewnego rodzaju komitet, który zamiast prostego podliczania wyników głosowania stosuje dodatkowy model adaptacyjny. Podejście to uzyskało dobrze obrazującą je nazwę *stacking*.

2.14 Problem niepełności danych

Częstym problemem w analizie danych jest niepełność dostępnych opisów. Systemy klasyfikacji najskuteczniej są w stanie uczyć się na danych, które są pełne, tj. każdy wektor ma określoną wartość w każdym wymiarze. W praktyce jednak nie jest to możliwe.

Powód niepełności opisu wektora może być różny. W najogólniejszym spojrzeniu można przyczyny podzielić na dwie grupy: braków całkowicie losowych oraz nieprzypadkowych. Całkowicie losowe braki mogą wynikać np. z zagubienia danych czy niedopatrzenia osoby zbierającej dane, która była w posiadaniu odpowiedniej informacji, ale nie dopełniła formalności wpisania jej w odpowiednie miejsce. Nieprzypadkowe braki w danych mogą być spowodowane tym, że z jakiejś przyczyny dany pomiar nie został dokonany. Na przykład w medycynie można nie wykonać pacjentowi pewnych badań, bo z innych wykonanych badań wynika, że nie jest to konieczne. Innym przykładem jest baza danych z wynikami w nauce osiąganymi przez studentów – wyników może brakować, bo dany student zrezygnował z dalszej nauki itp.

Te dwie grupy braków w danych powinny być całkiem inaczej traktowane przez uczące się systemy. Nieprzypadkowe braki niosą ze sobą pewną informację i ta informacja może być wykorzystywana w procesie uczenia. Jeśli jednak braki są całkowicie przypadkowe, to ich korelacja z etykietami klas może być również przypadkowa i z takiej informacji nie należy korzystać.

Nie wszystkie modele adaptacyjne mają możliwości uwzględniania braków w danych. Kiedy systemy wymagają pełnych opisów wektorów, wówczas najczęstszym podejściem jest wpisanie w miejsce braków pewnych wyróżniających je wartości – np. takich, które nie należą do dziedziny danej cechy. To w jakich sytuacjach i w jaki sposób można te braki wypełniać jest kwestią bardzo delikatną. Nie można, na przykład, wypełniać braków pewną unikalną wartością, by proces adaptacji z niej korzystał, kiedy brak ten jest nieprzypadkowy, bo ta unikalna wartość może się doskonale korelować z klasą i wówczas uwzględnienie tego w budowanym modelu powoduje, że dostaniemy model bardzo dokładny, ale bezużyteczny.

Bardzo często w miejsca brakujących wartości wpisywane są średnie wartości występujące w zbiorze danych dla poszczególnych cech. Można też spotkać rozwiązania polegające na wypełnianiu braków średnimi wartościami dla danej cechy wśród wektorów z tej samej klasy co uzupełniany wektor. Jest to oczywiście metoda, która może wprowadzić w błąd system uczący się na tak przygotowanych danych. Najlepiej, jeśli braki są traktowane w ten sam sposób dla zbioru treningowego i testowego (by zachować modelowi podobne warunki testu jak te, które miał podczas uczenia), natomiast w tym przypadku nie ma możliwości wypełnienia braku w ten sam sposób dla wektora, którego klasy nie znamy (a jeśli

ją znamy, to nie potrzebujemy do jego klasyfikacji żadnego modelu inteligencji obliczeniowej).

Systemy klasyfikacji, które bazują na liczeniu odległości pomiędzy wektorami, czasami mierzą odległości w przestrzeni zredukowanej do tych wymiarów, które są znane dla obu argumentów funkcji odległości. Nie jest to również metoda doskonała, ponieważ mnogość braków powoduje traktowanie danych wektorów jak bardzo bliskie sobie, co często jest sprzeczne z prawdą.

Najrozsądniejszym rozwiązaniem wydaje się być używanie zamiast odległości wartości oczekiwanych odległości. Odległość pomiędzy dwoma wektorami jest w istocie zmienną losową. W sytuacji, kiedy obydwa wektory sa w pełni określone, rozkład wartości odległości można uznać za jednopunktowy. Jeśli pewne wartości nie są znane, to rozkład ten zależy od rozkładów poszczególnych brakujących wartości. Przy założeniu reprezentatywności próbki danych treningowych (bez którego próby rozwiązania zadania klasyfikacji są niezasadne), rozkłady brakujących wartości można uznać za tożsame z rozkładem znanych wartości dla danej cechy w zbiorze treningowym. Skoro odległość jest zmienną losową, to tam, gdzie oczekujemy konkretnej wartości najtrafniejszym jest użycie wartości oczekiwanej tej zmiennej.

Przykład 2.5 *Załóżmy, że dysponujemy zbiorem treningowym* $T = \{(t^i, c^i) : i = 1,\ldots,n\} \subseteq X \times C$ *i chcemy zmierzyć odległość pomiędzy wektorami x i y przestrzeni X. Niech* K_i *będzie zbiorem tych indeksów* $1,\ldots,n$*, dla których wartość* t_i *jest znana. Jeśli nie znamy wartości i-tej cechy wektora x, ale znamy i-tą cechę wektora y, to i-ty składnik sumy liczonej dla zmierzenia wartości oczekiwanej odległości euklidesowej powinien być równy*

$$E(x_i - y_i)^2 = \frac{1}{|K_i|} \sum_{j \in K_i} (t_i^j - y_i)^2. \tag{2.60}$$

Jeśli nie znamy również i-tej współrzędnej wektora y, to wówczas

$$E(x_i - y_i)^2 = \frac{1}{|K_i|^2} \sum_{(j,k) \in K_i^2} (t_i^j - t_i^k)^2. \tag{2.61}$$

Warto zauważyć, że jeśli x_i *oraz* y_i *są znane, to*

$$E(x_i - y_i)^2 = (x_i - y_i)^2. \tag{2.62}$$

Oczywiście takie naliczanie odległości ma skutek całkowicie inny niż liczenie odległości po wpisaniu średnich w miejsce braków, bo np. jeśli szacujemy odległość od kompletnie znanego wektora do całkowicie nieznanego, to zamiast 0

mamy wartość oczekiwaną odległości od znanego wektora (średnią po wektorach zbioru treningowego), co jest najsłuszniejszym oszacowaniem tej odległości.

Jeśli na temat danego wektora, w którym brakuje pewnych opisów, mamy pewną dodatkową wiedzę, to można ją oczywiście wykorzystać w oszacowaniu wartości oczekiwanej. W szczególności można, na przykład, liczyć średnie nie dla odległości od wszystkich wektorów zbioru treningowego, a tylko od pewnej liczby najbliższych (w odpowiednio zredukowanej przestrzeni) sąsiadów. Można też ograniczać uśrednianie do danych z tej samej klasy co analizowany wektor, ale trzeba mieć na względzie, że nie można tego samego oszacowania użyć dla danych testowych.

Rozdział 3

Kryterium separowalności i jego zastosowania

Metody klasyfikacji takie jak drzewa decyzji dążą do osiągnięcia swego celu przez stopniowe oddzielanie od siebie próbek z różnych klas. Jakość oddzielania oceniana jest w różnych systemach na różne sposoby (najczęściej z wykorzystaniem teorii informacji).

Algorytmy prezentowane w tym rozdziale opierają się na prostym kryterium, które próbuje w bardziej naturalny sposób (niż metody teorii informacji) oceniać separowalność obiektów z różnych klas. W uproszczeniu separowalność może być zdefiniowana jako liczba par obiektów należących do różnych klas, które zostały odseparowane wskutek danego podziału.

3.1 Kryterium i jego własności

Ponieważ cechy opisujące obiekty mogą być ciągłe bądź dyskretne, a te dwa rodzaje mają różną „naturę", należy w systemach klasyfikacji traktować różne rodzaje cech w różny sposób. Aby zdefiniować separowalność dla danego podziału musimy najpierw zdefiniować *wartości podziału* (ang. *split value*). W literaturze dotyczącej drzew decyzji najczęściej mówi się o *punkcie podziału* (ang. *split point*) lub *punkcie odcięcia* (ang. *cut-off point*), jednak pojęcie „wartość" jest bardziej ogólne i lepiej obejmuje także przypadki cech dyskretnych.

Definicja 3.1 ***Wartością podziału dla cechy ciągłej*** *jest dowolna liczba rzeczywista.* ***Wartością podziału dla cechy dyskretnej*** *jest dowolny podzbiór zbioru możliwych wartości danej cechy. Wartość podziału nazywa się również krótko* ***podziałem****.*

Dla ułatwienia definicji separowalności warto wprowadzić pojęcia *lewej i prawej strony podziału*.

Definicja 3.2 ***Lewą stroną*** *podziału s dla cechy F i danego zbioru obiektów D nazywamy*

$$\mathsf{LS}(s,F,D) \;=\; \begin{cases} \{x \in D : F(x) < s\} & \textit{jeśli } F \textit{ jest ciągła} \\ \{x \in D : F(x) \in s\} & \textit{w przeciwnym wypadku} \end{cases} \tag{3.1}$$

gdzie $F(x)$ jest wartością cechy F dla wektora x.
Prawą stroną *podziału s dla cechy F i danego zbioru obiektów D nazywamy*

$$\mathsf{RS}(s,F,D) \;=\; D \setminus \mathsf{LS}(s,F,D) \tag{3.2}$$

A zatem lewa strona podziału danego zbioru dla cechy ciągłej, to podzbiór tych elementów tego zbioru, które mają wartość cechy f mniejszą niż wartość podziału. Nazwa *lewa strona* ma tutaj naturalną interpretację graficzną. Dla cech dyskretnych pojęcie to jest wprowadzone sztucznie jako podzbiór tych obiektów, które dla cechy f przyjmują wartość należącą do zbioru będącego wartością podziału. W tym przypadku nazwa nie ma już naturalnych skojarzeń, ale taka definicja znacznie upraszcza ostateczną definicję separowalności.

Definicja 3.3 ***Separowalnością podziału*** *s dla cechy F i zbioru obiektów D (ang.* Separability of Split Value – SSV*) nazywamy*

$$\begin{aligned} \mathsf{SSV}(s,F,D) \;=\; & 2 \cdot \sum_{c \in C} |\mathsf{LS}(s,F,D_c)| \cdot |\mathsf{RS}(s,F,D \setminus D_c)| \\ & - \sum_{c \in C} \min(|\mathsf{LS}(s,F,D_c)|, |\mathsf{RS}(s,F,D_c)|) \end{aligned} \tag{3.3}$$

gdzie C jest zbiorem klas a D_c jest zbiorem wektorów z D, które należą do klasy c.

Za optymalny podział dla danej cechy uznajemy taki, który maksymalizuje separowalność. W praktyce najczęściej nie ma przeszkód, by przeanalizować wszystkie możliwe wartości podziału. Dla cech dyskretnych wymaga to sprawdzenia wszystkich podzbiorów zbioru symboli danej cechy, więc jest to wykonalne pod warunkiem, że wartości cechy nie jest zbyt wiele. Jeśli cecha ma więcej niż kilkanaście możliwych wartości, wówczas warto ograniczyć analizowane podzbiory np. do podzbiorów o mocy nie większej niż pewna wartość dobrana tak, aby analizie poddawać stosowną liczbę podzbiorów. System SSV ogranicza liczbę sprawdzanych podzbiorów tak, by nie była ona dużo większa niż liczba wektorów – z jednej strony ogranicza to złożoność, a z drugiej zabezpiecza przed nieuzasadnionym faworyzowaniem cech dyskretnych nad ciągłe (problem ten jest mocno akcentowany w [120]). W przypadku cech ciągłych istotnie różne separowalności

możemy uzyskać tylko dla podziałów pomiędzy którymi leży przynajmniej jeden wektor z separowanych danych. Oznacza to, że wystarczy ocenić separowalność punktów leżących w połowie odległości pomiędzy sąsiednimi wartościami cech występującymi w danych (twierdzenie 3.3 na stronie 76 pozwala na dalszą redukcję zbioru sprawdzanych podziałów). Jeśli mimo to otrzymamy kilka sąsiednich punktów podziału o tej samej separowalności, to za optymalny najlepiej uznać ich średnią arytmetyczną, bo to maksymalizuje prawdopodobieństwo, że podział jest dobry nie tylko dla separowanych właśnie danych, ale i dla innych danych wygenerowanych z tego samego rozkładu.

Pierwszy człon wyrażenia definiującego separowalność ma tutaj zasadnicze znaczenie i wydaje się być dość naturalnym sformalizowaniem potocznego rozumienia pojęcia separowalności. Drugi człon jest dodatkiem, który ma znaczenie wtedy, gdy istnieje wiele podziałów o tej samej wartości pierwszego członu i promuje te podziały, które jak najmniej separują pary wektorów reprezentujących tę samą klasę. W praktyce sytuacja, w której wiele punktów podziału daje tę samą separowalność ma miejsce zwykle w problemach wieloklasowych (dla więcej niż dwóch klas). Podobne zadanie w innych systemach klasyfikacji pełni binaryzacja (p. rozdział 3). W przypadku kryterium separowalności można również rozpatrywać nieco inną jego postać, a mianowicie:

$$\begin{aligned} \mathsf{SSV}'(s,F,D) &= \Big(\sum_{c\in C} |\mathsf{LS}(s,F,D_c)| \cdot |\mathsf{RS}(s,F,D\setminus D_c)|, \\ &\quad - \sum_{c\in C} |\mathsf{LS}(s,F,D_c)| \cdot |\mathsf{RS}(s,F,D_c)| \Big) \end{aligned} \tag{3.4}$$

Przy takiej definicji separowalność to para liczb, a porównywanie separowalności to sprawdzanie porządku leksykograficznego (druga współrzędna ma znaczenie dopiero wówczas, gdy pierwsze współrzędne są równe).

Uwaga : Zaprezentowane dwie definicje separowalności nie są równoważne. W zastosowaniach dają różne wyniki, jednak różnice są na tyle nieznaczne, że w praktyce wystarczy stosować dowolnie wybraną z nich.

Lemat 3.1 *Niech X będzie przestrzenią klasyfikacji, F jedną z cech tej przestrzeni, C pewnym zbiorem klas oraz $D \subseteq X \times C$ skończonym zbiorem takim, że $\forall_{c\in C} D_c \neq D$.*

1. *Jeżeli s jest podziałem cechy F takim, że $\mathsf{LS}(s,F,D) \neq \emptyset$ oraz $\mathsf{RS}(s,F,D) \neq \emptyset$, to $\mathsf{SSV}(s,F,D) > 0$.*

2. *Jeżeli s jest podziałem cechy F to $\mathsf{SSV}(s,F,D) = 0$ wtedy i tylko wtedy, gdy $\mathsf{LS}(s,F,D) = \emptyset$ lub $\mathsf{RS}(s,F,D) = \emptyset$.*

Dowód : Własność 1 łatwo udowodnić indukcyjnie ze względu na moc zbioru D.

1. $|D| = 2$
   Z założeń wynika, że D składa się z dwóch elementów, z których każdy należy do innej klasy, i które są rozdzielone podziałem s. Stąd $\mathsf{SSV}(s,F,D) = 2 > 0$.

2. $|D| > 2$
   Z założeń wynika, że

$$\exists_{x_1,x_2 \in D} \; x_1 \in \mathsf{LS}(s,F,D) \wedge x_2 \in \mathsf{RS}(s,F,D) \wedge C(x_1) \neq C(x_2) \tag{3.5}$$

   oraz, że przynajmniej jeden ze zbiorów $D \setminus \{x_1\}$ i $D \setminus \{x_2\}$ spełnia warunki Lematu dla zbioru D. Bez utraty ogólności możemy założyć, że pierwszy z tych zbiorów spełnia te warunki. Z założenia indukcyjnego mamy więc $\mathsf{SSV}(s,F,D \setminus \{x_1\}) > 0$. Zauważmy, że

$$\mathsf{SSV}(s,F,D) = \mathsf{SSV}(s,F,D \setminus \{x_1\}) + 2 \cdot \sum_{c \in C \setminus \{C(x_1)\}} |\mathsf{RS}(s,F,D_c)| - a \tag{3.6}$$

   gdzie w przypadku gdy $|\mathsf{LS}(s,F,D \setminus \{x_1\})| < |\mathsf{RS}(s,F,D \setminus \{x_1\})|$ mamy $a = 1$, a w przeciwnym przypadku $a = 0$. Stąd oraz w związku z tym, że $\sum_{c \in C \setminus \{C(x_1)\}} |\mathsf{RS}(s,F,D_c)| \geqslant 1$ mamy tezę.

Dostateczność warunku z punktu 2 Lematu wynika wprost z definicji, konieczność jest konsekwencją punktu 1. □

Twierdzenie 3.2 *Przy założeniach Lematu 3.1:*
Jeżeli

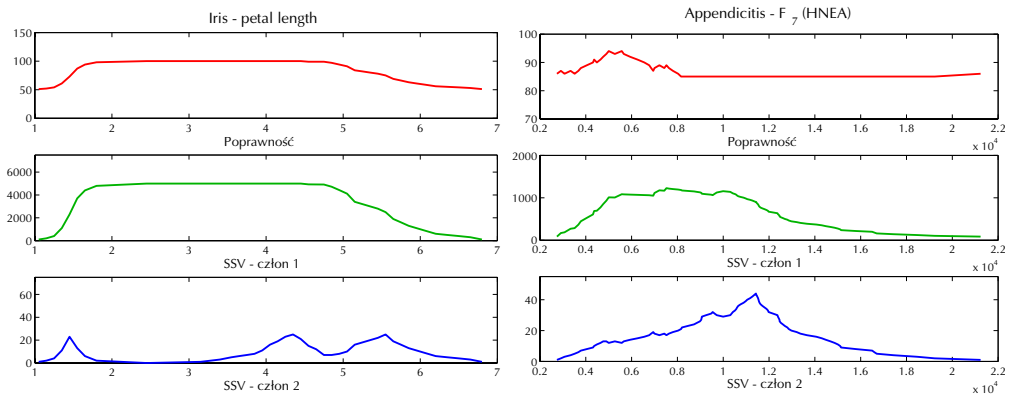
$$|\{F(x) : x \in D\}| > 1 \tag{3.7}$$

(cecha F wśród elementów zbioru D przyjmuje przynajmniej dwie różne wartości), to dla podziału s_0 cechy F takiego, że $s_0 = \mathsf{argmax}_s \; \mathsf{SSV}(s,F,D)$ mamy

$$\mathsf{LS}(s_0,F,D) \neq \emptyset, \quad \mathsf{RS}(s_0,F,D) \neq \emptyset \quad \textit{oraz} \quad \mathsf{SSV}(s_0,F,D) > 0 \tag{3.8}$$

Dowód : Wynika wprost z lematu 3.1. □

Powyższe twierdzenie opisuje bardzo korzystną z praktycznego punktu widzenia własność kryterium, jaką jest gwarancja istnienia podziału poprawiającego separowalność różnych klas, jeśli tylko zbiór, który chcemy rozdzielić, zawiera dane z przynajmniej dwóch klas oraz cecha według której chcemy zbiór podzielić przyjmuje w danym zbiorze przynajmniej dwie różne wartości.

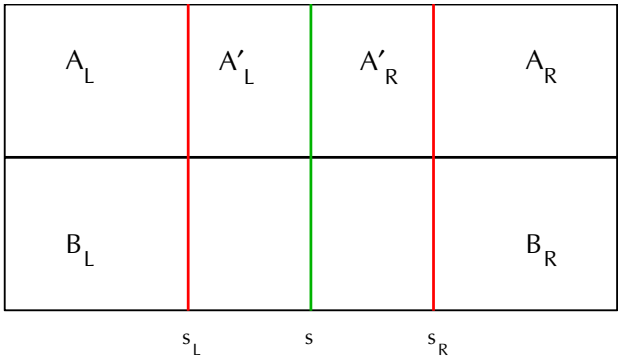


Rysunek 3.1: Przykład wykresów wartości kryterium SSV dla wybranych cech zbiorów *Iris* i *Appendicitis*. Wykres górny przedstawia poprawność klasyfikacji przy danym podziale, środkowy – główną część kryterium SSV, dolny – dodatkową część kryterium. Wykres pełnej wartości kryterium SSV byłby optycznie nierozróżnialny od jego głównej części (drugi człon ma inny rząd wielkości).

Wykres obrazujący wartość kryterium separowalności dla różnych wartości podziału dla cechy ciągłej charakteryzuje się tym, że poza zakresem wartości reprezentowanych przez dzielony zbiór przyjmuje wartość 0, a wewnątrz tego zakresu wartości dodatnie. Przykład takiego wykresu przedstawia rysunek 3.1. Oznacza to, że kryterium SSV pozwala dzielić każdy zbiór danych rekurencyjnie tak długo, aż otrzymamy podzbiory, w których albo wszystkie wektory będą należały do tej samej klasy, albo będą nierozróżnialne.

Kryterium SSV nadaje się do oceny separowalności zarówno dla cech ciągłych jak i dyskretnych. Co więcej oceny dla różnych cech można między sobą porównywać otrzymując w ten sposób informację, która z cech daje lepsze rozróżnienie klas.

Możliwych wartości podziałów dla cechy ciągłej jest oczywiście nieskończenie wiele. W praktycznych przypadkach wystarczy jednak analiza znacznie ograniczonego ich zbioru. Przede wszystkim trzeba zauważyć, że dla danego zbioru treningowego aby rozpatrzyć wszystkie wartości podziału, dla których kryterium SSV może dać różne oceny, wystarczy wziąć punkty dzielące (najlepiej w połowie przedziału) sąsiednie wartości reprezentowane w tym zbiorze treningowym. Wystarczy zatem posortować wektory treningowe według danej cechy, a potem analizować podziały algorytmem o złożoności liniowej względem liczby wektorów – górnym ograniczeniem liczby podziałów do przeanalizowania jest liczba wektorów zbioru treningowego. Twierdzenie 3.3 pozwala jeszcze bardziej zredukować zbiór wartości podziałów niezbędnych do oceny – mówi o tym, że można



Rysunek 3.2: Ilustracja dowodu twierdzenia 3.3

pominąć wartości podziału umiejscowione pomiędzy wektorami reprezentującymi tę samą klasę.

Twierdzenie 3.3 *Niech X będzie przestrzenią klasyfikacji, F – jedną z cech tej przestrzeni, C – pewnym zbiorem klas, $c \in C$, $T \subseteq X \times C$ – zbiorem treningowym. Jeśli $(x,c) \in T$, $(y,c) \in T$, $x_F < y_F$ oraz $\forall_{z \in T} z_F \in [x_F, y_F] \rightarrow C(z) = c$, to żadna wartość podziału z przedziału (x_F, y_F) nie maksymalizuje wartości kryterium* SSV.

Dowód : Przyjmijmy następujące oznaczenia:

$$A_L = \{t \in T : t_F < x_F \wedge C(t) = c\}$$

$$A_R = \{t \in T : t_F > y_F \wedge C(t) = c\}$$

$$B_L = \{t \in T : t_F < x_F \wedge C(t) \neq c\}$$

$$B_R = \{t \in T : t_F > y_F \wedge C(t) \neq c\}$$

Niech s_L oraz s_R będą wartościami podziału takimi, że $s_L < x_F$ i $\mathsf{LS}(s_L, F, T) = A_L \cup B_L$ oraz $s_R < x_F$ i $\mathsf{RS}(s_R, F, T) = A_R \cup B_R$. Rozpatrujemy wszystkie podziały $s \in [s_L, s_R]$ i oznaczamy:

$$A'_L = \{t \in T : t_F \in [x_F, s)\}$$

$$A'_R = \{t \in T : t_F \in [s, y_F]\}$$

$$A' = A'_L \cup A'_R$$

Opisaną sytuację przedstawia rysunek 3.2. Oznaczmy moce zbiorów odpowiada-

jącymi im małymi literami (tj. $|A_L| = a_L$ itp.). Ponieważ

$$\begin{aligned}\mathsf{SSV}(s,F,T) &= 2\cdot((a_L+a'_L)b_R+b_L(a_R+a'_R)) \\ &\quad -\min(a_L+a'_L,a_R+a'_R)-\sum_{d\in C}\min(d_L,d_R),\end{aligned} \tag{3.9}$$

gdzie d_L i d_R to stosowne liczebności pozostałych klas (stałe przy zadanej zmienności s), więc maksymalizacja kryterium, to maksymalizacja wyrażenia

$$2\cdot(a'_L\cdot b_R+b_L\cdot a'_R)-\min(a_L+a'_L,a_R+a'_R) \tag{3.10}$$

przy ograniczeniach $a'_L \geqslant 0$, $a'_R \geqslant 0$ oraz $a'_L + a'_R = a'$ czyli wyrażenia

$$2\cdot a'_L(b_R-b_L)-\min(a_L+a'_L,a_R+a'-a'_L) \tag{3.11}$$

przy ograniczeniu $a'_L \in [0,a']$.

Wystarczy pokazać, że SSV osiąga maksimum wtedy i tylko wtedy, gdy $a'_L = 0$ albo $a'_L = a'$. Rozpatrzmy następujące przypadki:

$b_R = b_L$ Wówczas SSV osiąga maksimum jeśli $\min(a_L + a'_L, a_R + a' - a'_L)$ jest minimalne, czyli w zależności od relacji pomiędzy a_L a a_R w punkcie 0 lub a'.

$b_R > b_L$ Wówczas SSV osiąga maksimum dla $a'_L = a'$, bo ze wzrostem a'_L o 1 pierwszy składnik wzrasta o 2 a drugi maleje o 1.

$b_R < b_L$ Wówczas SSV osiąga maksimum dla $a'_L = 0$ z analogicznych powodów jak w poprzednim przypadku.

Zatem w każdym z przypadków SSV osiąga maksimum wtedy i tylko wtedy, gdy $a'_L = 0$ albo $a'_L = a'$. □

Analogiczną własność do określonej twierdzeniem 3.3 posiada również kryterium SSV′.

SSV a krzywe ROC. Można zauważyć pewne podobieństwa łączące kryterium separowalności SSV z ideą krzywych ROC. Maksymalizacja pola pod krzywą dla klasyfikatora dyskretnego jest równoważna maksymalizacji sumy wrażliwości i znamienności, którą można przedstawić jako:

$$\frac{\mathsf{TP}(C_i,f,T)\cdot\mathsf{Neg}(C_i,T)+\mathsf{TN}(C_i,f,T)\cdot\mathsf{Pos}(C_i,T)}{\mathsf{Pos}(C_i,T)\cdot\mathsf{Neg}(C_i,T)}. \tag{3.12}$$

Ponieważ mianownik jest wartością stałą, maksymalizacji podlega sam licznik, który można również przedstawić (upraszczając nieco zapisy) jako

$$2\cdot\mathsf{TP}\cdot\mathsf{TN}+\mathsf{TP}\cdot\mathsf{FN}+\mathsf{TN}\cdot\mathsf{FP}. \tag{3.13}$$

Pierwszy składnik jest dokładnie taki sam jak pierwszy człon kryterium SSV dla przypadku zadania dwuklasowego. Pozostałe składniki istotnie różnią te dwa wyrażenia, a więc mogą prowadzić do różnych ekstremów obu funkcji (choć oczywiście są zbieżne w przypadku pełnej dokładności klasyfikacji, która jest nadrzędnym celem).

SSV a inne rozwiązania. Podobne spojrzenie na problem separowalności danych leży u podstaw prac grupy prof. Bobrowskiego, która używa *kryterium dipolowego* do konstruowania sieci neuropodobnych [12, 13, 15]. Idea ta polega na konstruowaniu sieci poprzez dodawanie kolejnych warstw ukrytych. Każda warstwa tworzona jest przez sekwencyjne dodawanie kolejnych neuronów i optymalizację wag połączeń wejściowych poprzez minimalizację jednego z dwóch proponowanych kryteriów, które podobnie jak SSV bazują na określaniu liczb rozseparowanych i nie rozseparowanych par wektorów należących do różnych klas (*dipole mieszane*) oraz do tej samej klasy (*dipole czyste*). Minimalizacji wartości kryteriów dokonuje się z pomocą algorytmów genetycznych bądź metod *wymiany rozwiązań bazowych* [11, 14], które są bliskie metodom programowania liniowego.

Inne podobne kryterium jest używane w systemie CART – *Gini index*. Nieczystość węzłów jest tutaj także zależna wprost od liczby nie rozseparowanych par wektorów. Jest jednak wyrażona w języku rachunku prawdopodobieństwa, a maksymalizacji poddawana jest miara przyrostu czystości 2.44, która wprowadza dodatkowe ważenie stosownymi wartościami prawdopodobieństw, co może prowadzić do rozwiązań istotnie różnych niż uzyskiwane z użyciem kryterium SSV.

3.2 Drzewa decyzji

Kryterium określone definicją 3.3 i jego własności sprawiają, że jego naturalnym zastosowaniem jest budowanie drzew klasyfikacji. Pozwala ono porównać ze sobą różne podziały dla tej samej cechy, a nawet dla różnych cech, więc daje możliwość wyboru maksymalnie dobrego (w sensie tego kryterium) podziału. Wystarczy w sposób hierarchiczny wybierać kolejne maksymalnie dobre podziały, aby w ten sposób otrzymać drzewo decyzji, które z maksymalną dokładnością separuje od siebie elementy należące do różnych klas (jeśli tylko w danym problemie klasyfikacji, dla którego budowane jest drzewo, nie występują wektory wzajemnie sprzeczne tj. identyczne wektory przypisane do różnych klas, to można zbudować drzewo klasyfikujące zbiór treningowy z dokładnością 100%).

Drzewa decyzji oparte na kryterium SSV to drzewa binarne – kolejne podziały przestrzeni, realizowane przez podwęzły, to podziały na lewą i prawą stronę stosownej wartości podziału. Nie stosuje się tu binaryzacji, ponieważ kryterium,

oceniając zdolność separowania obiektów z różnych klas, jest niezależne od decyzji związanych z węzłami.

Niektóre zastosowane metody szukania drzew wymagają porównywania nie tylko separowalności podziałów, ale także całych drzew. Dlatego przydatne jest wprowadzenie pojęcia separowalności dla całych drzew klasyfikacji.

Definicja 3.4 ***Separowalnością drzewa*** *D dla danego zbioru $T \subseteq X$ nazywamy sumę separowalności podziałów odpowiadających wszystkim węzłom drzewa, które nie są liśćmi. Oznaczamy ją przez* $\mathsf{SSV}(D)$.

Warto zauważyć, że maksymalna separowalność drzewa oznacza nie tylko maksymalną poprawność klasyfikacji drzewa, ale zarazem (dzięki drugiemu członowi definicji 3.3) także w pewnym sensie najprostszą strukturę drzewa.

Aby sprawniej opisywać algorytmy budowania drzew wprowadźmy pojęcia *liścia finalnego* drzewa i *drzewa finalnego*:

Definicja 3.5 *Liść drzewa jest* ***liściem finalnym*** *dla zbioru treningowego T jeśli wszystkie wektory zbioru T, które wpadają do tego liścia należą do tej samej klasy lub są nierozróżnialne.*

Drzewo nazywamy ***finalnym*** *dla zbioru treningowego T jeśli wszystkie jego liście są finalne.*

3.2.1 Metody szukania

Zbiór wszystkich możliwych drzew stanowi przestrzeń, która choć zwykle jest zdecydowanie mniejsza niż przestrzeń wszystkich klasyfikatorów dla danego problemu, jest na ogół na tyle duża, że pełne jej przeszukanie jest niemożliwe. Ponieważ nie jest też możliwe analityczne wyznaczenie optymalnego drzewa decyzji, więc najskuteczniejszym dla wyznaczenia drzewa klasyfikacji wydaje się użycie pewnych metod szukania heurystycznego. Oczywiście różne metody szukania mogą (choć nie muszą) prowadzić do różnych drzew klasyfikacji.

Jedną z najprostszych metod, które nadają się do tego zadania jest metoda szukania o nazwie *najpierw najlepszy* (ang. *best first search*, w skrócie BFS).

Algorytm 3.1 (Szukanie drzewa metodą „najpierw najlepszy")

► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$.*

◄ ***Wynik:*** *Binarne drzewo klasyfikacji.*

1. *Budujemy drzewo składające się z pojedynczego liścia o stałej funkcji przynależności o wartości* 1 *oraz etykiecie klasy dominującej w zbiorze T (jeśli więcej niż jedna klasa ma maksymalną liczebność w T, to wybieramy arbitralnie jedną z nich).*

2. *Jeśli każdy z liści drzewa jest liściem finalnym, to kończymy algorytm.*

3. *Znajdujemy liść, który nie jest liściem finalnym. Oznaczmy go symbolem L.*

   (a) *Niech T_L będzie zbiorem tych wektorów zbioru T, które wpadają do L.*

   (b) *Wyznaczamy wartości kryterium* SSV *dla wszystkich podziałów wynikających z rozkładu danych w T_L:*

      i. *Dla każdej cechy dyskretnej oceniamy wszystkie możliwe podziały (jeśli ich liczba jest większa niż $2|T_L|$, to rozpatrujemy tylko podzbiory o mocy nie większej niż wartość dobrana tak, by liczba tych podzbiorów była jak najbliższa $|T_L|$).*

      ii. *Dla każdej cechy ciągłej oceniamy podziały, które odpowiadają średnim dla sąsiednich wartości (reprezentowanych w zbiorze T_L) tej cechy, z pominięciem tych podziałów, które na mocy twierdzenia 3.3 nie maksymalizują wartości kryterium* SSV.

   (c) *Wybieramy podział s i cechę F o maksymalnej wartości kryterium* SSV *dla zbioru T_L i dodajemy do drzewa dwa podwęzły węzła L o funkcjach przynależności $\mathbf{1}_{\mathsf{LS}(s,F,T_L)}$ oraz $\mathbf{1}_{\mathsf{RS}(s,F,T_L)}$ i etykietach odpowiadających klasom dominującym odpowiednio w $\mathsf{LS}(s,F,T_L)$ oraz $\mathsf{RS}(s,F,T_L)$.*

4. *Wracamy do punktu 2.*

Uwaga : Praktyczna realizacja analizy wszystkich istotnie różnych podziałów dla cechy ciągłej polega na posortowaniu danych według wartości tej cechy, a następnie analizie wszystkich średnich arytmetycznych dla par sąsiadujących w uporządkowanej liście wektorów, które mają różne wartości.

Specyfika drzew klasyfikacji sprawia, że algorytm szukania najpierw najlepszy ma nieco prostszą niż klasyczna postać. Nie ma tutaj potrzeby zapamiętywania odwiedzonych w procesie szukania stanów w celu ewentualnego powrotu, ponieważ w każdym następnym kroku dostajemy drzewo o lepszej separowalności niż w kroku poprzednim, stąd nowy stan drzewa na pewno będzie najlepszym spośród dotąd analizowanych. Mamy więc tutaj do czynienia z metodą przypominającą przeszukiwanie w głąb tyle, że stosującą heurystyki do określania, którą gałąź należy najpierw rozwinąć, czyli metodą *wspinaczkową* (ang. *hill climbing*).

Analiza złożoności obliczeniowej algorytmu 3.1 jest przedstawiona w rozdziale 3.2.4.

Bardziej kosztowną obliczeniowo metodą szukania również zastosowaną do szukania drzew opartych na kryterium SSV jest metoda *szukania wiązką* (ang. *beam search* – BS).

Algorytm 3.2 (Szukanie drzewa wiązką)

► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$ oraz szerokość wiązki $w \in N$.*

◄ ***Wynik:*** *Binarne drzewo klasyfikacji, bądź zestaw wiązek.*

1. *Niech $B_0 = \{D_0\}$, gdzie D_0 jest drzewem złożonym z pojedynczego liścia o stałej funkcji przynależności o wartości 1 oraz etykiecie klasy dominującej w T.*
2. *$i \leftarrow 0$*
3. *Tak długo, jak B_i nie zawiera drzewa finalnego:*
   (a) *$i \leftarrow i+1$*
   (b) *Niech $B_i = \emptyset$.*
   (c) *Dla każdego drzewa D w B_{i-1}:*
      i. *Dla każdego z liści L drzewa D:*
         A. *Dla każdego podziału s każdej cechy F wynikającego z rozkładu zbioru T_L wektorów z T wpadających do L:*
            - *Liczymy separowalność $\mathsf{SSV}(D_s)$ drzewa klasyfikacji D_s otrzymanego z D przez dodanie podwęzłów liścia L o funkcjach przynależności $\mathbf{1}_{\mathsf{LS}(s,F,X)}$ oraz $\mathbf{1}_{\mathsf{RS}(s,F,X)}$ i etykietach odpowiadających klasom dominującym odpowiednio w zbiorach $\mathsf{LS}(s,F,T_L)$ oraz $\mathsf{RS}(s,F,T_L)$.*
            - *Jeśli $|B_i| < w$ to*
              $$B_i \leftarrow B_i \cup \{D_s\}.$$
              W przeciwnym przypadku:
              jeśli $\mathsf{SSV}(D_s) > \min_{D' \in B_i} \mathsf{SSV}(D')$, to
              $$B_i \leftarrow B_i \setminus \{\operatorname*{argmin}_{D' \in B_i} \mathsf{SSV}(D')\} \cup \{D_s\}$$
            - *Jeśli D_s jest drzewem finalnym, to przechodzimy do punktu 4.*
4. *Wynikiem algorytmu jest drzewo finalne należące do B_i bądź lista wiązek $B_0, \ldots, B_i$.*

Wynikiem szukania wiązką może być pojedyncze drzewo finalne bądź cała historia procesu szukania w postaci listy wiązek, która może być poddawana analizie mającej na celu znalezienie drzewa najlepiej uogólniającego problem.

Drobna modyfikacja algorytmu szukania najpierw najlepszy pozwala sterować dokładnością przeszukiwań za pomocą zadanego parametru. Wystarczy zamiast oceny jakości podziału przez prostą ocenę jego separowalności (jak w punkcie 2 algorytmu 3.1) zastosować ocenę separowalności drzewa, które można zbudować korzystając z tego podziału i rozbudowując w nim poddrzewo o zadanej głębokości. Dla głębokości równej 1 otrzymamy algorytm równoważny algorytmowi 3.1, a dla odpowiednio dużego parametru głębokości otrzymamy algorytm pełnego przeszukiwania przestrzeni drzew:

Algorytm 3.3 (Metoda najpierw najlepszy z parametrem głębokości)

► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$ oraz maksymalna głębokość analizowanego poddrzewa $g \in N$.*

◄ ***Wynik:*** *Binarne drzewo klasyfikacji.*

1. *Budujemy drzewo składające się z pojedynczego liścia o stałej funkcji przynależności o wartości* 1 *oraz etykiecie klasy dominującej w zbiorze T (jeśli więcej niż jedna klasa ma maksymalną liczebność w T, to wybieramy arbitralnie jedną z nich).*

2. *Jeśli każdy z liści drzewa jest liściem finalnym, to kończymy algorytm.*

3. *Znajdujemy liść, który nie jest liściem finalnym. Oznaczmy go symbolem L.*

   (a) *Niech T_L będzie zbiorem tych wektorów zbioru T, które wpadają do L.*

   (b) *Dla każdej z cech F i dla każdego z podziałów s cechy F wynikających z rozkładu danych w T_L:*

      i. *Dodajemy do drzewa dwa podwęzły węzła L o funkcjach przynależności $\mathbf{1}_{\mathsf{LS}(s,F,X)}$ oraz $\mathbf{1}_{\mathsf{RS}(s,F,X)}$ i etykietach odpowiadających klasom dominującym odpowiednio w $\mathsf{LS}(s,F,T_L)$ oraz $\mathsf{RS}(s,F,T_L)$.*

      ii. *Jeśli $g > 1$ to wykonujemy dwa razy rekurencyjnie algorytm dla parametrów $T = \mathsf{LS}(s,F,T_L), g = g-1$ oraz $T = \mathsf{RS}(s,F,T_L), g = g-1$, a drzewa będące rezultatami dołączamy jako odpowiednie poddrzewa w węzłach dodanych w punkcie 3(b)i.*

      iii. *Zapamiętujemy separowalność poddrzewa zbudowanego w liściu L jako ocenę jakości podziału s.*

      iv. *Usuwamy wszystkie węzły dodane w punktach 3(b)i oraz 3(b)ii.*

(c) *Wybieramy podział s i cechę F o maksymalnej jakości (wyznaczonej w punkcie 3(b)iii) i dodajemy do drzewa dwa podwęzły węzła L o funkcjach przynależności $\mathbf{1}_{\mathsf{LS}(s,F,X)}$ oraz $\mathbf{1}_{\mathsf{RS}(s,F,X)}$ i etykietach odpowiadających klasom dominującym odpowiednio w $\mathsf{LS}(s,F,T_L)$ oraz $\mathsf{RS}(s,F,T_L)$.*

4. *Wracamy do punktu 2.*

W praktyce duże wartości parametru g prowadzą do nieakceptowalnie długich czasów wykonania. Wartość parametru $g = 2$ jest często maksymalną którą daje się zastosować w realnych problemach.

Intuicje każą przypuszczać, że większy koszt szukania wiąże się z lepszym drzewem decyzji. W myśl zasady minimalnej długości opisu należy przewidywać, że im prostsze będzie drzewo, tym lepszą uzyskamy generalizację. Ponieważ szukanie wiązką i zmodyfikowana metoda szukania typu „najpierw najlepszy" są nieco dokładniejsze niż szukanie typu najpierw najlepszy, teoretycznie mamy większą szansę znaleźć mniej złożone drzewo. Jednak w praktyce nie zawsze wiąże się to z lepszymi wynikami w testach generalizacji. Efekt ten dostrzegli i dyskutowali Quinlan i Cameron-Jones w [153]. Nie podają oni formalnego uzasadnienia takiego stanu rzeczy, a tylko potwierdzają swoje spostrzeżenia doświadczeniami oraz sugerują, że przyczyna może leżeć w tym, że dokładne szukanie pozwala znaleźć rozwiązania, które bardzo dobrze pasują do zbioru treningowego, ale nie mają zdolności generalizacji. Jednak, jeśli znajdowane rozwiązania mają strukturę nie bardziej skomplikowaną niż te wykrywane prostszymi metodami, to ich brak generalizacji byłby w sprzeczności z zasadą minimalnej długości opisu. Wydaje się więc, że problem może wiązać się z tym, że przy dokładniejszym szukaniu trudniej jest weryfikować zdolności generalizacyjne modeli. Ponieważ przy drobnych zmianach w zbiorze treningowym można spodziewać się znaczących różnic pomiędzy modelami, trudno jest na podstawie znalezionego dla pewnej próbki danych modelu wnioskować o własnościach innego, zbudowanego dla innego zbioru danych. Modele znajdowane prostszymi metodami szukania, choć również w takiej sytuacji mogą być różne, to jednak mają większą szansę dzielenia pewnych własności (z powodu zbliżonego sposobu ich uzyskania). Innym powodem nieintuicyjnych relacji wyników może być fakt, że heurystyki stosowane podczas szukania, są zbieżne z funkcją celu tylko w skrajnych przypadkach, więc minimalizacja błędu klasyfikacji i optymalizacja wartości danego kryterium nie są tożsame, a zatem w sytuacjach wymagających skomplikowanych modeli dokładniejsze szukanie znajduje atrakcyjniejsze wartości kryteriów, które mogą się wiązać z mniej dokładnymi modelami. Procesy dokładniejszego szukania prowadzą w inne (bardziej atrakcyjne z punktu widzenia danego kryterium) obszary przestrzeni modeli niż proste metody, a to nie zawsze oznacza atrakcyjniejsze wartości poprawności

klasyfikacji. Potwierdzeniem tych spostrzeżeń może być fakt, że w przypadkach, kiedy dokładniejsze metody budują mniej skuteczne drzewa, to modele te nie są mniej złożone niż modele znajdowane prostymi metodami.

W zastosowaniu do niektórych zadań klasyfikacji metody dokładniejszego szukania mogą przynieść interesujące rezultaty nawet dla binarnych drzew w których na każdym poziomie wybierany jest podział maksymalizujący poprawność klasyfikacji zbioru treningowego. Taka strategia daje dobre wyniki tylko wtedy, gdy w każdym węźle istnieje podział poprawiający jakość klasyfikacji. Spektakularnym przykładem są tutaj dane *Mushroom* (wyniki dla tych danych przedstawione są na stronie 101). Jeśli dane nie pozwalają znaleźć podziału, który zmniejsza błąd klasyfikacji (tak jest bardzo często np. wtedy, gdy jedna z klas zdecydowanie dominuje w zbiorze treningowym), należy oczekiwać bardzo słabych wyników tej prostej metody.

3.2.2 Generalizacja

Drzewo wygenerowane jednym z algorytmów poprzedniego podrozdziału klasyfikuje zbiór treningowy z maksymalną dokładnością. Zazwyczaj oznacza to nadmierne dopasowanie modelu do danych treningowych, czyli brak generalizacji. Struktura drzewa charakteryzuje się tym, że główny węzeł realizuje najbardziej ogólne rozróżnienie klas, jego podwęzły bardziej szczegółowe, aż do najbardziej szczegółowych decyzji determinowanych przez liście, które często rozdzielają między sobą nawet pojedyncze wektory z różnych klas. Naturalnym jest więc, że w celu uzyskania z pełnego drzewa bardziej uogólnionej reprezentacji wiedzy należy *obciąć* te liście[1], które są nadmiernie szczegółowe. Wystarczy więc znaleźć kryterium oceny, które z liści należy usunąć, by uzyskać drzewo optymalnie generalizujące.

Przycinanie do zadanego stopnia. Jedną z metod przycinania drzewa zastosowanych do drzew decyzji opartych na kryterium SSV jest przycinanie *do zadanego stopnia*. Obcięcie drzewa do stopnia n oznacza usunięcie wszystkich liści, których dodanie do drzewa spowodowało poprawę jakości klasyfikacji o nie więcej niż n wektorów.

Algorytm 3.4 (Przycinanie drzewa do zadanego stopnia)

- ► ***Dane:*** *Drzewo klasyfikacji D, zbiór treningowy T, stopień przycinania $n \in N$.*
- ◄ ***Wynik:*** *Zmodyfikowane drzewo D.*

[1] Proces ten nazywa się także *przycinaniem* lub *oczyszczaniem* drzewa (od ang. *prune*).

Powtarzamy:

1. *liczba usuniętych* $\leftarrow 0$
2. *Dla każdego węzła W, który jest nadwęzłem dwóch liści drzewa D (oznaczmy te liście przez W_1 oraz W_2):*
   (a) *Wyznaczamy wartość $G(W) = E(W) - E(W_1) - E(W_2)$, gdzie $E(X)$ oznacza liczbę wektorów ze zbioru T, które wpadają do węzła X, ale należą do innej klasy niż etykieta węzła X.*
   (b) *Jeśli $G(W) \leqslant n$, to usuwamy węzły W_1 oraz W_2 z drzewa D (węzeł W staje się liściem) oraz liczba usuniętych $\leftarrow$ liczba usuniętych + 1.*

tak długo jak liczba usuniętych $\neq 0$.

Przycinanie do zadanej liczby liści drzewa. Inną metodą zastosowaną do drzew SSV jest przycinanie drzewa do zadanej liczby liści, która polega na usuwaniu najmniej atrakcyjnych (z punktu widzenia poprawności klasyfikacji) podziałów aż do uzyskania drzewa o zadanej liczbie liści.

Algorytm 3.5 (Przycinanie drzewa do zadanej liczby liści)

► ***Dane:*** *Drzewo klasyfikacji D, zbiór treningowy T, docelowa liczba liści $n \in N$.*

◄ ***Wynik:*** *Zmodyfikowane drzewo D.*

Tak długo jak liczba liści drzewa D jest większa niż n powtarzamy:

1. *Dla każdego węzła W, który jest nadwęzłem dwóch liści drzewa D (oznaczmy te liście przez W_1 oraz W_2) wyznaczamy wartość $G(W) = E(W) - E(W_1) - E(W_2)$.*
2. *Spośród węzłów spełniających założenia poprzedniego punktu wyznaczamy węzeł W^0 o minimalnej wartości $G(W^0)$.*
3. *Usuwamy węzły W_1^0 oraz W_2^0 z drzewa D (węzeł W^0 staje się liściem).*
4. *Przycinamy drzewo D do stopnia* 0.

Uwagi :

1. Algorytmy opierają się na założeniu, że podziały, które powodują najmniejszą (bądź wręcz żadną) poprawę jakości klasyfikacji zbioru treningowego są najmniej wartościowe z punktu widzenia generalizacji (zwiększają złożoność modelu nie poprawiając przy tym jakości klasyfikacji).

2. Przycięcie drzewa do stopnia 0 (p. algorytm 3.4) to usunięcie tych liści, które nie poprawiają klasyfikacji swego nadwęzła.

3. Przycięcie drzewa do liczby liści równej *n* może powodować, że liczba liści w drzewie będzie mniejsza niż *n*. Dzieje się tak za sprawą punktu 4, który usuwa dodatkowo liście, które nie poprawiają (w stosunku do swojego nadwęzła) klasyfikacji zbioru treningowego.

4. Przycinanie jest oparte na usuwaniu węzłów, które mają najmniejszy wkład do poprawności klasyfikacji, a nie do separowalności mierzonej kryterium SSV. Jednym z powodów jest fakt, że kryterium SSV zostało zaprojektowane do porównywania różnych podziałów tego samego zbioru, i nie nadaje się do porównywania podziałów zbiorów różnej mocy. Najważniejszym powodem jest jednak brak konieczności używania heurystyk zastępujących ocenę poprawności klasyfikacji, która może być tutaj wykorzystana bez żadnych przeszkód.

Optymalizacja przycinania. Stopień przycinania i docelowa liczba liści drzewa są dodatkowymi parametrami algorytmu tworzenia drzew decyzji. Dobór tych parametrów można jednak zautomatyzować tak, by system sam dbał o optymalną złożoność drzewa, a więc i jego zdolności generalizacyjne. Podstawową metodą zastosowaną dla tego celu jest tzw. *uczenie przez kroswalidację*:

Algorytm 3.6 (Szukanie optymalnego drzewa przez kroswalidację)

► ***Dane:*** *Zbiór treningowy T, liczba przebiegów kroswalidacji $n \in N$, typ kroswalidacji, rodzaj przycinania drzewa.*

◄ ***Wynik:*** *Binarne drzewo klasyfikacji.*

1. *Dzielimy zbiór treningowy na n części dokładnie tak jak w zwykłej kroswalidacji (p. rozdział 2.2.3) otrzymując n par zbiorów* (T_i^{TRN}, T_i^{TST}) .

2. *Dla każdego* $i \in \{1, \ldots, n\}$*:*

   (a) *Budujemy drzewo klasyfikacji* D_i *używając* T_i^{TRN} *jako zbioru treningowego.*

   (b) *Dla każdej wartości stopnia przycinania (od* 0 *do* $|T|$*) albo docelowej liczby liści (od* 0 *do* $|T|$*) liczymy błąd walidacyjny, czyli błąd klasyfikacji zbioru* T_i^{TST}.

3. *Sumujemy błędy walidacyjne dla danej wartości parametru dla poszczególnych przebiegów kroswalidacji.*

4. *Wyznaczamy wartość parametru p o najniższej sumie błędów.*

5. *Budujemy drzewo decyzji D dla zbioru treningowego* T.

6. *Przycinamy drzewo D do stopnia albo liczby liści p.*

7. *Drzewo D jest wynikiem algorytmu.*

Uwagi :

1. Za błąd walidacyjny można wziąć ważoną sumę $E_{WAL} = E_{TRN} + \mathrm{n} \cdot E_{TST}$, gdzie E_{TRN} to błąd klasyfikacji naszego modelu dla T_i^{TRN} a E_{TST} to błąd dla T_i^{TST}. Zabezpiecza to przed ekstremalnym obcięciem drzewa na użytek mniejszego błędu na części testowej, które może mieć miejsce zwłaszcza w przypadku małych zbiorów danych. Jednak dla większych zbiorów taki błąd walidacyjny ma tendencję do pozostawiania nadmiernie rozbudowanego drzewa.

2. Zamiast jednego parametru (definiującego sposób przycinania drzewa – poziom lub docelową liczbę liści) mamy teraz dwa parametry: liczbę przebiegów kroswalidacji oraz typ kroswalidacji. Mimo to taki algorytm jest mniej zależny od ustawień użytkownika, bo parametry kroswalidacji wywierają zdecydowanie mniejszy wpływ na końcowe efekty.

3. W przypadku uczenia przez kroswalidację część zbioru używaną do wewnętrznego testu dokładności nazywa się raczej częścią *walidacyjną* a nie testową (dla odróżnienia od rzeczywistych testów modelu).

3.2.3 Indukcja reguł

Drzewa decyzji oparte na kryterium SSV dają się w bardzo łatwy sposób przedstawić w równoważnej postaci reguł logiki klasycznej pierwszego rzędu. Każda gałąź drzewa reprezentuje jedną regułę, której poprzednik jest koniunkcją przesłanek wyznaczonych przez funkcje przynależności do poszczególnych węzłów, a następnik stwierdzeniem przynależności wektora do klasy będącej etykietą liścia tej gałęzi. W takim ujęciu reguły wygenerowane z drzewa obejmują obszary

przestrzeni, które są rozłączne oraz pokrywają całą przestrzeń. Dla każdej klasy można więc stworzyć pojedynczą regułę, która będzie alternatywą koniunkcji pewnej liczby wyrażeń atomowych (innymi słowy będzie w postaci normalnej).

W przypadku bardziej złożonych drzew, często okazuje się, że dla niektórych liści, pewne decyzje podjęte w drzewie na poziomie bliższym korzeniowi nie mają wpływu na klasyfikację danych ze zbioru treningowego. Można więc uprościć reguły im odpowiadające usuwając te z przesłanek, które nie mają pozytywnego wpływu na wynik treningowy otrzymując w ten sposób prostsze reguły, które są równie dokładne jak (a w rzadkich przypadkach mogą być nawet bardziej dokładne niż) drzewo, z którego zostały wygenerowane. Jest to podejście bardziej uzasadnione niż sposób generowania reguł przyjęty w metodzie C4.5 (p. rozdział 2.9.3), gdzie niezależna optymalizacja każdej z reguł często powoduje „przesuwanie" reguł w kierunku większych skupisk danych, co owocuje dużą utratą dokładności klasyfikacji (nawet o 10% i więcej [188]).

3.2.4 Analiza systemu

Algorytm budowania drzew z wykorzystaniem kryterium SSV, podobnie do większości algorytmów drzew decyzji, a także do metod minimalnoodległościowych i wielu innych metod heurystycznych jest metodą nieparametryczną. Metody heurystyczne takie jak drzewa decyzji oparte na kryterium SSV czy kryteriach wywodzących się z teorii informacji, a także metody minimalnoodległościowe, nie opierają się na jakichkolwiek założeniach co do rozkładów danych w przestrzeni klasyfikacji – nie estymują one gęstości rozkładów, ale wprost konstruują stosowną funkcję decyzyjną. W związku z tym drzewo SSV może dobrze uogólniać informacje zawarte w danych, nawet w przypadku rozkładów o nieskończonej wariancji czy niezdefiniowanej wartości oczekiwanej. Na przykład, problem klasyfikacji zbioru liczb do dwóch klas: liczb dodatnich i ujemnych, może być bardzo skutecznie rozwiązywany już na podstawie małych próbek danych treningowych.

W pełni formalne przedstawienie założeń przy których heurystyczne drzewo decyzji gwarantuje dobrą generalizację jest bardzo trudne [131]. W przypadku dowolnego klasyfikatora podstawowym założeniem jego skuteczności jest *reprezentatywność* zbioru treningowego dla całej populacji – innymi słowy zapewnienie, że w danych treningowych jest zawarta odpowiednia informacja na temat zadania klasyfikacji. Precyzyjne określenie tej reprezentatywności nie jest możliwe, ponieważ jest to pojęcie mocno zależne od charakteru danego zadania klasyfikacji. Dla drzew SSV jest to jedyne założenie konieczne, czego uzasadnieniem może być powyższy przykład zadania klasyfikacji liczb dodatnich i ujemnych. Trudno jest wskazać założenia dostateczne, które by zasadniczo nie ograniczały dziedziny zastosowań.

Pewnym potwierdzeniem słuszności strategii budowania drzew decyzji może

być fakt, że algorytmy te dążą do indukcji jak najprostszych struktur o jak najwyższej poprawności. Jest więc to droga bardzo bliska zasadzie minimalnej długości opisu, która ma solidne podstawy teoretyczne (p. rozdział 2.2.2).

W rzeczywistych problemach klasyfikacji, zwykle mamy mocno ograniczone możliwości tworzenia próbek danych, co sprawia, że na ogół, nie jesteśmy w stanie spełnić warunków optymalności, na których oparte są dowody uniwersalności systemów. Dlatego też, dla potencjalnych zastosowań, cenniejszym jest empiryczne potwierdzenie skuteczności danej metody.

Złożoność algorytmów. Dla poniższych oszacowań złożoności obliczeniowej algorytmów przyjmijmy następujące oznaczenia:

- n – liczba wektorów zbioru danych,
- c – wymiar przestrzeni klasyfikacji,
- k – liczba klas,
- v – liczba wartości symbolicznych danej cechy,
- w – liczba węzłów drzewa.

Wprawdzie, liczba węzłów drzewa w jest wynikiem działania algorytmu, a nie parametrem wejściowym, jednak jej oszacowanie w zależności od parametrów wejściowych jest niemożliwe. Dla bardzo dużych wartości n oraz c, w może pozostawać bardzo małe. Oszacowaniem pesymistycznym jest $w = n$, choć tylko w wyjątkowo trudnych problemach liczba węzłów drzewa może przekroczyć $\log n$. Szacowanie złożoności z wykorzystaniem zmiennej w pozwala uwzględnić czynnik interpretowany jako „trudność rozwiązywanego problemu".

Wybór podziału maksymalizującego kryterium SSV dla cechy ciągłej polega na posortowaniu wektorów według wartości tej cechy ($O(n\log n)$) a następnie oszacowaniu wartości SSV dla każdego z możliwych podziałów (pesymistycznie $O(nk)$, choć twierdzenie 3.3 mówi, że ta złożoność może być mniejsza). Ostatecznie mamy więc oszacowanie $O(n\log n + nk)$.

Dla cechy dyskretnej wybór podziału maksymalizującego SSV polega na wyliczeniu liczebności poszczególnych klas ($O(n)$), następnie wartości SSV dla każdego z podzbiorów zbioru możliwych wartości cechy ($O(2^v vk)$). W praktyce dla zmniejszenia złożoności obliczeniowej (w wypadku dużych v), oraz dla uniknięcia mocno akcentowanego w [120] problemu niesłusznego preferowania podziałów cech dyskretnych względem cech ciągłych, można ograniczyć liczbę rozpatrywanych podzbiorów do n (cel prostoty opisu dyktuje pomysł analizowania podzbiorów o możliwie małej mocy). Otrzymujemy zatem złożoność $O(n + \min(n, 2^v)vk) \leqslant O(nvk)$.

Szukanie drzewa metodą „najpierw najlepszy" wymaga wyznaczenia optymalnego podziału dla w węzłów. Im węzeł jest dalej od korzenia, tym mniej danych analizuje – wszystkie węzły o tej samej „odległości" od korzenia analizują w

sumie (maksymalnie) n wektorów. Pesymistyczne oszacowanie liczby poziomów wynosi w, choć w praktyce, liczba ta przyjmuje wartości zdecydowanie bliższe $\log w$. Niemniej złożoność metody szukania maksymalnego drzewa należy ocenić jako $O((n\log n + nk + \min(n, 2^v)vk)cw)$.

Ocena jakości odpowiednio przycinanych drzew jest wykonywana przez test poprawności klasyfikacji różnych obcięć dla danego zbioru walidacyjnego. Zatem, pierwszym krokiem oceny jest sprawdzenie liczb błędów popełnianych przez każdy z węzłów ($O(n\log w)$).

Ocena jakości obcięć drzewa do wszystkich możliwych poziomów czy też liczb liści wymaga odpowiedniego uporządkowania węzłów oraz liniowego (względem liczby węzłów) wyliczenia poprawności. Otrzymujemy więc $O(n\log w + w\log w)$, czyli w istocie $O(n\log w)$. Ponieważ jest to mniejsza złożoność niż samego konstruowania drzewa, więc można ją pominąć.

Jeśli dla uproszczenia przyjmiemy, że wartości liczby klas k oraz wartości cech symbolicznych v, są mniejsze niż $\log n$, to otrzymamy proste oszacowanie całości algorytmu $O(ncw\log^2 n)$. W pesymistycznej ocenie otrzymujemy zatem $O(n^2 c\log^2 n)$, należy jednak uwzględnić, że często „trudność problemu", którą oznaczyliśmy przez w jest bliższa $\log n$ a nawet $\log\log n$.

3.2.5 Rezultaty

Zaprezentowane metody oparte na kryterium separowalności SSV zostały przetestowane na wielu dostępnych bazach danych. Wyniki dla różnych baz danych zostały zestawione poniżej. Rzetelne porównanie wyników z dostępnymi w literaturze nie jest łatwe, ponieważ różni autorzy używają różnych technik przygotowywania danych i zbierania wyników, i sposoby te nie zawsze są precyzyjnie opisane, więc trudno dokonać testów w podobnych warunkach. Często publikowane są uśrednione wyniki, ale bez odchyleń standardowych, albo nie jest jasno sprecyzowane czego one dotyczą (czy jest to średnia odchyleń standardowych wyników wewnętrznych kroswalidacji, czy odchylenie standardowe średnich wyników kroswalidacji).

Tabelaryczne zestawienia przedstawiają średnie poprawności klasyfikacji. Kiedy oceniamy drzewa decyzji albo systemy regułowe, istotnymi są także liczebność i czytelność reguł opisujących dane. Nawet jeśli w testach sprawdzających generalizację wyniki nie są statystycznie tożsame z najlepszymi, jakie daje się osiągnąć innymi systemami, to warto przyjrzeć się opisom regułowym, ponieważ mogą one zawierać w sobie sporo cennych informacji na temat danego zbioru danych.

Szczególnie bogatym źródłem informacji, które można wykorzystać w testach systemów klasyfikacji jest repozytorium Uniwersytetu Kalifornijskiego w Irvine (University of California at Irvine – UCI) [126]. Można tam znaleźć między innymi część zbiorów danych, dla których wyniki zostały zebrane w ramach projektu

STATLOG i przedstawione w [130].

Sporo wyników dla danych z UCI zawierają również prace Lim i in. [119, 118]. Przedstawione tam wyniki niestety nie zawierają wartości standardowych odchyleń błędów. Dla większych zbiorów danych autorzy stworzyli dla testu zbiory treningowy i testowy zawierające po 1000 wektorów, co może mieć bardzo istotny wpływ na uzyskiwane wyniki. Dla niektórych drzew decyzji (np. QUEST i CART) podane są wyniki dla dwóch metod przycinania (0-SE i 1-SE – p. rozdział 2.9.1). W ogólnych porównaniach systemów należałoby rozpatrywać je jako wyniki dwóch różnych systemów.

Duże ilości wyników dla danych z UCI zebrał również Zarndt [188]. Niestety sposób w jaki otrzymał część z liczb również nie pozwala na rzetelne porównania metod. Na przykład dla drzew C4.5 wykonywał testy z oczyszczaniem drzew oraz bez oczyszczania i umieścił w tabelach lepszy z uzyskanych wyników. Dla sieci neuronowych wykonał wielokrotne testy z różnymi ustawieniami parametrów i umieścił w pracy najlepsze spośród otrzymanych rezultatów. Należy więc te wyniki uznać raczej za maksima a nie za wartości oczekiwane poprawności klasyfikacji.

W przedstawionych poniżej zestawieniach pojawiają się także wyniki zaczerpnięte z pracy Štera i Dobnikara [169], a także z [165], [137], [8] i kilku innych. Przedstawiane liczby to wyrażone w procentach średnie poprawności klasyfikacji uzyskane w 10-krotnym teście kroswalidacyjnym bądź w teście na wydzielonym pliku testowym.

W porównaniach pojawiają się wyniki następujących metod:

AC^2 – interakcyjny system ekspertowy wykorzystujący drzewa na wzór NewID
ALLOC80 – analiza dyskryminacyjna
AQ15 – system indukcji reguł (p. rozdział 2.10.1)
ASI – drzewo decyzji [169]
ASR – drzewo decyzji [169]
Bayes Tree – Bayesowskie podejście do drzew decyzji [28]
Cal5 – drzewo decyzji (p. rozdział 2.9.5)
CART – drzewo decyzji (p. rozdział 2.9.1)
CASTLE – Causal Sructures From Inductive Larning
C4.5 – drzewo decyzji (p. rozdział 2.9.3)
C-MLP2LN – konstruktywistyczna wersja MLP2LN (p. rozdział 4.1
CN2 – system indukcji reguł (p. rozdział 2.10.2)
DIPOL92 – hybrydowy klasyfikator kawałkami liniowy
FDA – Analiza Dyskryminacyjna Fishera
FOIL – First Order Inductive Learning

FSM – sieć neuronowa Feature Space Mapping [50, 1]
FACT – drzewo decyzji FACT (p. rozdział 2.9.4)
GTO DT – Global Tree Optimization
GTS – General-to-Specific [89, 88]
IB – Instance Based Learning - kilka różnych metod (1-4)
IND – pakiet kilku różnych metod: bayes, bayes opt, mml, mml opt
IndCART – system CART w implementacji pakietu IND
ID3 – drzewo decyzji (p. rozdział 2.9.2)
IncNet – ontogeniczna sieć neuronowa [98]
ITRULE – system indukcji reguł (p. rozdział 2.10.3)
kNN – metoda k najbliższych sąsiadów
Kohonen – Sieci Kohonena (uczone bez nadzoru)
LDA – Liniowa Analiza Dyskryminacyjna
LERS – Learning from Examples based on Rough Sets [80, 81]
LFC – drzewo decyzji [169]
LMDT – Linear Machine Decision Tree (p. rozdział 2.9.6)
LogDA – dyskryminacja liniowa z użyciem funkcji logistycznych
LVQ – Learning Vector Quantizers (wersja z nadzorem)
MLP BP – sieci MLP z propagacją wsteczną błędu (p. rozdział 2.8)
MML – Minimum Message Length - drzewo decyzji
Naive Bayes – Naiwny Klasyfikator Bayesowski (p. rozdział 2.5)
NEFCLASS – system reguł rozmytych [137, 139]
NewID – drzewo decyzji (p. rozdział 2.9.6)
OC1 – drzewo decyzji Oblique Classifier
1R – reguły klasyfikacji wykorzystujące jedna cechę (p. rozdział 2.10)
PVM – Predictive Value Maximization [182]
QDA – Kwadratowa Analiza Dyskryminacyjna
QUEST – drzewo decyzji (p. rozdział 2.9.4)
RBF – Radial Basis Functions networks - sieci neuronowe
RIAC – Rule Induction through Approximate Classification - metoda indukcji reguł wykorzystująca teorię zbiorów przybliżonych
SBM – Similarity Based Methods [79]
SMART – statystyczny algorytm klasyfikacji i regresji
S-MLP – Search-Based MLP (p. rozdział 4.2)
S-Plus – jedna z implementacji systemu CART
SVM – Support Vector Machines (p. rozdział 2.7)

Część z tych metod została przedstawiona w [130]. W tabelach z wynikami przedstawione są źródła, z których zaczerpnięto wyniki. Napis KIS[2] oznacza, że wyniki zostały uzyskane w naszym zespole. Przy niektórych wynikach mogą pojawiać się dodatkowe napisy, których znaczenia są następujące:

auto k – kNN z automatyczną selekcją wartości k

k=1:10 – j.w. z k z zakresu od 1 do 10

komb.lin. – drzewo decyzji z użyciem przesłanek wykorzystujących kombinacje liniowe oryginalnych cech

mix – drzewo decyzji z użyciem zarówno kombinacji liniowych jak i pojedynczych zmiennych

Manh. – użytą miarą odległości była metryka miast (ang. *city block* lub *Manhattan*)

0-SE – drzewo obcinane metodą 0-SE (p. rozdział 2.9.1)

1-SE – drzewo obcinane metodą 1-SE (p. rozdział 2.9.1)

std – dane poddano standaryzacji przed użyciem metody

***n*xCV –** wynik pochodzi z testu CV o liczbie przebiegów równej *n*

Wyniki otrzymane przy pomocy systemu SSV w zdecydowanej większości zostały uzyskane dla domyślnych parametrów metody (tzn. z obcinaniem do optymalnej liczby liści wyznaczonej uczeniem wewnętrzną 10-krotną kroswalidacją). Wyniki uzyskane z innymi parametrami są odpowiednio oznaczone.

Iris

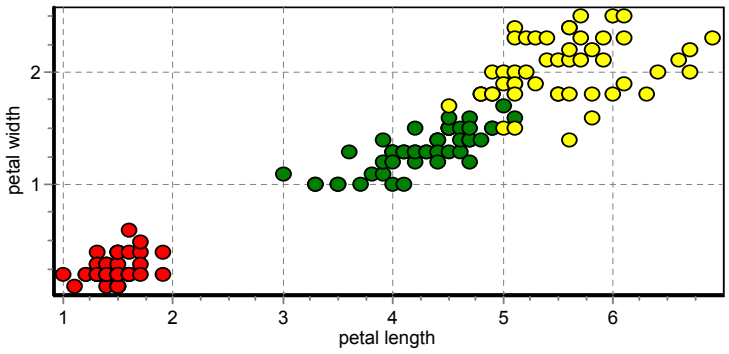
Zbiór danych *Iris* służy zwykle jako jeden z podstawowych testów systemów klasyfikacji jako, że jest jednym z tych, o których strukturze wiadomo już wszystko. Zawiera 150 wektorów opisujących kwiaty irysa czterema pomiarami (długości i szerokości listków kielicha i płatków). Zadanie polega na rozróżnieniu od siebie trzech klas irysów: *setosa*, *virginica*, *versicolor*.

Do uzyskania w pełni zadowalającego wyniku wystarczy użyć cechy trzeciej i czwartej tj. *petal length* oraz *petal width*. Rzut wektorów na podprzestrzeń, którą te dwa wymiary wyznaczają, przedstawia rysunek 3.3. Oddzielenie klasy *setosa* od pozostałych jest zadaniem trywialnym dla każdego systemu klasyfikacji. Rozdzielenie pozostałych dwóch klas, przy zachowaniu zdolności generalizacji, wiąże się z popełnieniem od 3 do 9 błędów (poprawność odpowiednio 98% i 94%).

Drzewo SSV dla zbioru *Iris* opisują następujące reguły:

$\mathcal{R}_1$: petal length $< 2.45 \rightarrow$ *setosa*

[2]Katedra Informatyki Stosowanej Uniwersytetu Mikołaja Kopernika w Toruniu



Rysunek 3.3: Dane *Iris* w rzucie na dwa z czterech wymiarów

$\mathcal{R}_2$: petal length $\in (2.45, 4.95) \wedge$ petal width $< 1.65 \rightarrow$ *virginica*
$\mathcal{R}_3$: *else versicolor*

Odtwarzają one klasyfikację zbioru z dokładnością 98%. Taką samą dokładność oferuje inny zestaw reguł (uzyskany dla innych parametrów modelu), który w kontekście badanego zbioru obiektów realizuje dokładnie tę samą funkcję klasyfikującą:

$\mathcal{R}_1$: petal width $< 0.8 \rightarrow$ *setosa*
$\mathcal{R}_2$: petal width $\in (0.8, 1.65) \wedge$ petal length $< 4.95 \rightarrow$ *virginica*
$\mathcal{R}_3$: *else versicolor*

Opisy te potwierdzają skuteczność metody drzew klasyfikacji SSV, wykazując zgodność wiedzy wydobytej przez drzewa z wiedzą uzyskaną wieloma innymi metodami.

Appendicitis

Zbiór danych *Appendicitis* został udostępniony przez Sholoma Weissa [182]. Zawiera 106 wektorów opisujących wyniki testów medycznych różnych pacjentów. Każdy przypadek opisany jest przez 7 pomiarów oznaczonych nazwami: WBC1, MNEP, MNEA, MBAP, MBAA, HNEP i HNEA, oraz ma przypisaną jedną z dwóch klas.

Drzewo decyzji otrzymane dla tego zbioru danych systemem SSV, po konwersji do postaci reguł logicznych, przybiera postać czterech reguł (po dwie na klasę). Ponieważ drzewo decyzji nie udziela odpowiedzi *nie wiem* (tj. dzieli całą przestrzeń klasyfikacji na dwie części i każdemu wektorowi przypisuje jedną z etykiet), pary reguł odpowiadające klasom wzajemnie się uzupełniają, a więc pełnym opisem drzewa jest tutaj zestaw dwóch reguł dla jednej z klas poszerzony o warunek *w przeciwnym przypadku* (ang. *else*).

$\mathcal{R}_1$: HNEA $< 7520.5 \wedge$ MBAP $< 12 \rightarrow$ *class 1*
$\mathcal{R}_2$: HNEA $\in (9543.5, 9997.5) \rightarrow$ *class 1*
$\mathcal{R}_3$: *else class 2*

Taki zestaw reguł opisuje zbiór z poprawnością klasyfikacji 94.3%. Pomijając drugą regułę, która obejmuje trzy wektory nie objęte pierwszą, uzyskujemy dokładność 91.5%.

Jest to najprostszy ze znanych dotąd opisów tego zbioru danych. Biorąc pod uwagę wyniki różnych systemów, można z dużym prawdopodobieństwem stwierdzić, że nie istnieje prostszy i dokładniejszy od tego opis.

Tabela 3.1 przedstawia wyniki różnych systemów uzyskane uczeniem na całym zbiorze oraz w teście *leave-one-out*.

Metoda	Poprawność	Poprawność L10
PVM [182]	91.5%	89.6%
SSV– wiązką	94.3%	88.7%
SSV	94.3%	87.7%
RIAC	–	86.9%
MLP BP	90.2%	85.8%
CART	90.6%	84.9%
Naive Bayes	88.7%	83.0%
kNN	–	82.1%
C-MLP2LN, 1 neuron [42]	91.5%	
C-MLP2LN, 2 neurony	94.3%	

Tabela 3.1: Wyniki dla danych *Appendicitis*

Oprócz bardzo dokładnego i zwartego opisu jaki dostajemy ucząc drzewo SSV na całości danych, również w teście L1O daje jedną z wyższych (w porównaniu z innymi systemami) poprawność testową. Lepszy wynik uzyskuje jedynie algorytm PVM, który opiera się na pełnym przeszukiwaniu (co w przypadku tak małego zbioru danych jest możliwe), więc prawdopodobnie definiuje górną granicę błędu L1O jaki można uzyskać dla tych danych.

Dla metody C-MLP2LN przeprowadzenie testu L1O nie było możliwe ze względu na niezbędność interakcji użytkownika w procesie uczenia sieci (proces ten musiałby być przeprowadzony 106 razy).

Hypothyroid

Zbiór *Hypothyroid* pochodzi z repozytorium UCI [126]. Jest rozdzielony na dwie części dedykowane do używania odpowiednio jako zbiory treningowy i testowy.

W zbiorze treningowym znajdują się 3772 wektory, a w testowym 3428. Przestrzeń klasyfikacji jest 21-wymiarowa, przy czym 15 cech jest binarnych a pozostałe 6 to cechy ciągłe. Obiekty rozkładają się na trzy klasy (*primary hypothyroid*, *compensated hypothyroid* oraz *normal*), przy czym do ostatniej należy 3488 wektorów zbioru treningowego (tj. 92.47%).

Reguły odpowiadające drzewu SSV szukanemu wiązką dla zbioru treningowego przedstawiają się następująco:

$\mathcal{R}_1$: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid-surgery = 0 $\rightarrow$ *primary hypothyroid*

$\mathcal{R}_2$: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid-surgery = 0 $\wedge$ on-thyroxine = 0 $\wedge$ TT4 $<$ 150.5 $\rightarrow$ *compensated hypothyroid*

$\mathcal{R}_3$: *else normal*

Poprawności na zbiorach treningowym i testowym, to odpowiednio 99.79% oraz 99.33%. Podobnie jak w przypadku zbioru *Appendicitis* i na temat tego zbioru wiadomo już bardzo dużo i wiedza ta potwierdza bardzo wysoką dokładność tego zwartego opisu. Stosując adaptacyjny algorytm stopniowego wyżarzania (*Adaptive Simulated Annealing* – ASA) do optymalizacji zbioru reguł (p. rozdział 5.1), uzyskaliśmy [45] jedynie nieco lepsze od powyższego rozwiązanie.

Najlepszy ze znanych wyników daje heterogeniczna wersja drzewa SSV (rozdział 3.3). Niektóre z wyników dla zbiorów *Hypothyroid* przedstawia tabela 3.2.

Warto zauważyć, że systemy neuronowe (włącznie z MLP optymalizowanym metodami genetycznymi [160]) i korelacją kaskadową) dają błędy dwukrotnie i więcej razy większe niż drzewo SSV i inne z najlepszych wyników (PVM, CART czy C-MLP2LN). Różnice, które można obserwować pomiędzy najskuteczniejszymi dla tych danych metodami są wielokrotnościami 0.03%, która to wartość odpowiada różnicy jednego wektora. Oczywiście różnica jednego błędu na 3488 przypadków jest różnicą znikomą, więc wyniki o takich różnicach należy uznać za równoważne.

Jak widać, dane *Hypothyroid* dają się najtrafniej klasyfikować systemami, które mają interpretację regułową. Oznacza to, że granice decyzji, z którymi mamy w tym przypadku do czynienia są ułożone prostopadle do osi układu współrzędnych i są bardzo ostre, co utrudnia minimalizację sieciom neuronowym realizującym łagodne przejścia pomiędzy klasami. W takich przypadkach również metody minimalnoodległościowe okazują się bardzo nieskuteczne.

Ponieważ wyniki są pomiarami medycznymi, może to oznaczać, że reguły, które tutaj odkrywamy, ujawniają sposób, w jaki decyzję podejmowali eksperci, czyli lekarze – najprawdopodobniej brali pod uwagę kilka najistotniejszych wyników i posługiwali się określonymi dla nich normami. Być może w takich sytuacjach sami lekarze nie byliby w stanie precyzyjnie określić na jakiej zasadzie podejmowali diagnozę. Tym bardziej wartościowe są opisy regułowe, jeśli mogą

Metoda	Popr. tren.	Popr. test.	Źródło
Hetero SSV (rozdział 3.3)	99.84%	99.39%	
C-MLP2LN + ASA	99.89%	99.36%	KIS [45]
IndCART 1-SE	–	99.36%	Lim et al. [119, 118]
IndCART 0-SE	–	99.33%	Lim et al. [119, 118]
PVM	99.79%	99.33%	Weiss & Kapouleas [182]
SSV	99.79%	99.33%	
C4.5	–	99.2%	Zarndt [188]
QUEST 0-SE	–	99.12%	Lim et al. [119, 118]
CART	–	99.1%	Zarndt [188]
QUEST 1-SE	–	99.1%	Lim et al. [119, 118]
C-MLP2LN	99.68%	99.07%	KIS
ID3	–	98.7%	Zarndt [188]
Korelacja kaskadowa	100%	98.48%	KIS
MLP BP	99.60%	98.45%	KIS
Naive Bayes	97.03%	96.06%	KIS
kNN	–	95.27%	KIS
LDA	–	93.81%	Lim et al. [119, 118]
Cal5	–	92.74%	Lim et al. [119, 118]

Tabela 3.2: Wyniki treningowe i testowe dla danych *Hypothyroid*

być pomocą w uzmysłowieniu sobie rzeczywistych przyczyn takich a nie innych decyzji ekspertów.

Wisconsin breast cancer

Dane na temat raka piersi (*Wisconsin breast cancer*) zawierają 699 opisów przypadków, spośród których 241 (34.5%) sklasyfikowano jako nowotwory złośliwe (*malignant*) a 458 (65.5%) jako łagodne (*benign*). Każda z dziewięciu cech opisujących przypadki przyjmuje wartości całkowite od 1 do 10.

Reguły z drzewa SSV (szukanego wiązką) dla tych danych wyglądają następująco:

$\mathcal{R}_1$: $F_3 < 2.5 \wedge F_6 < 1.5 \rightarrow$ *benign*
$\mathcal{R}_2$: $F_3 < 2.5 \wedge F_1 < 5.5 \rightarrow$ *benign*
$\mathcal{R}_3$: $F_3 > 2.5 \wedge F_6 < 2.5 \wedge F_5 < 3.5 \rightarrow$ *benign*
$\mathcal{R}_4$: *else malignant*

Ich poprawność to 97.4%. Jest to bardzo zwarty opis i dokładniejszy niż najlepsze z wyników osiąganych w kroswalidacji (wyniki te przedstawia tabela 3.3).

Metoda	Poprawność	Odch.st.	Źródło
IncNet	97.1		Jankowski [98]
kNN (auto k)	96.8	0.3	KIS
FDA	96.8		Ster & Dobnikar [169]
Hetero SSV	96.7	0.2	
MLP BP	96.7		Ster & Dobnikar [169]
LVQ	96.6		Ster & Dobnikar [169]
FSM	96.5		KIS
Naive Bayes	96.4		KIS
SSV	96.3	0.2	
DB-CART	96.2		Shang [165]
LDA	96.0		Ster & Dobnikar [169]
QUEST 1-SE	95.9		Lim et al. [119, 118]
QUEST 0-SE	95.6		Lim et al. [119, 118]
LFC, ASI, ASR	94.4-95.6		Ster & Dobnikar [169]
MML	94.8	1.8	Zarndt [188]
C4.5-drzewo	94.7	2	Zarndt [188]
CART	93.5		Shang [165]
QDA	34.5		Ster & Dobnikar [169]

(a) Wyniki 10-krotnej kroswalidacji

Metods	Poprawność	Liczba reguł	Typ reguł
C-MLP2LN	99.0	5	klasyczne
C-MLP2LN	97.7	4	klasyczne
SSV	97.4	3	klasyczne
NEFCLASS [138]	96.5	2	rozmyte
NEFCLASS [137]	96.5	4	rozmyte
C-MLP2LN	94.9	2	klasyczne

(b) Wyniki systemów regułowych dla całego zbioru

Tabela 3.3: Wyniki dla danych *Wisconsin breast cancer*

Jednym ze skuteczniejszych systemów generujących zestawy reguł rozmytych jest NEFCLASS [137], który dane *Wisconsin breast cancer* opisał czterema regułami rozmytymi [139], zawierającymi przesłanki dotyczące wszystkich cech. Po zastosowaniu pewnych metod oczyszczania [138], otrzymano dwie reguły wykorzystujące pięć cech, przy zachowaniu dokładności wyjściowego zestawu reguł.

Cleveland heart disease

Zbiór danych określany nazwą *Cleveland heart disease* składa się z 303 wektorów sklasyfikowanych do dwóch klas: zdrowych (*healthy*) i chorych (*sick*). Każdy wektor opisany jest pięcioma atrybutami ciągłymi i ośmioma dyskretnymi.

Drzewo zbudowane dla całości danych przedstawiają następujące reguły:

$\mathcal{R}_1$: ca = 0.0 $\wedge$ (thal = 0 $\vee$ exang = 0) $\rightarrow$ *healthy*
$\mathcal{R}_2$: cp $\neq$ 2 $\wedge$ slope $\neq$ 2 $\rightarrow$ *healthy*
$\mathcal{R}_3$: *else sick*

Ich poprawność klasyfikacji to 85.8%. A zatem jest to jeden z najbardziej zwartych i zarazem dokładnych opisów tego problemu.

Klasyfikacja tego zbioru danych jest jednak problemem bardzo trudnym – przy małych zaburzeniach w zbiorze treningowym powstają drzewa SSV dość mocno różniące się od podanego. W efekcie w teście kroswalidacyjnym poprawność na zbiorze testowym jest zdecydowanie niższa niż poprawność powyższego zestawu reguł. Porównanie wyników kroswalidacji różnych systemów przedstawia tabela 3.4. Większość wyników pochodzi z testu 10-krotnej kroswalidacji. Niektóre były uzyskane w 5-krotnym teście, co zostało wyraźnie zaznaczone w tabeli.

Ljubljana breast cancer

Dane *Ljubljana breast cancer* składają się z 286 wektorów opisanych przez 9 cech. Część cech oryginalnie była ciągła, ale w zbiorze danych dostępne są wartości po dyskretyzacji. 201 opisów reprezentuje jedną z klas a pozostałe 85 drugą. Jest to bardzo trudny problem klasyfikacyjny, czego dowodzi fakt, że wiele systemów uzyskuje poprawności poniżej wartości bazowej (70.3%).

Drzewo SSV dla całości danych znajduje następującą prostą regułę:

$\mathcal{R}_1$: deg-malig $>$ 2.5 $\wedge$ inv-nodes $>$ 2.5 $\rightarrow$ *recurrence-events*
$\mathcal{R}_2$: *else no-recurrence-events*

Ta reguła klasyfikuje cały zbiór danych z poprawnością 76.2% przy wrażliwości 31.8% oraz znamienności 95.0%. Parametry te mówią wyraźnie, że jest to reguła obejmująca tylko najbardziej charakterystyczne przypadki raka piersi. Jest ona

Metoda	Poprawność	Odch.st.	Źródło
LDA	84.5%		Ster & Dobnikar [169]
FDA	84.2%		Ster & Dobnikar [169]
FSM	84.0%		KIS
kNN, k=1:10,Manh.(std)	83.8%	5.3	KIS– GM
Naive Bayes	82.5-83.4%		KIS, Ster & Dobnikar [169]
LVQ	82.9%		Ster & Dobnikar [169]
GTO DT (5xCV)	82.5%		Bennett & Blue [8]
SVM (5xCV)	81.5%		Bennett & Blue [8]
kNN	81.5%		Ster & Dobnikar [169]
MLP BP (std)	81.3%		Ster & Dobnikar [169], KIS
CART	80.8%		Ster & Dobnikar [169]
SSV	79.7%	1.1	
RBF (std)	79.1%		KIS (Tooldiag)
ASR	78.4%		Ster & Dobnikar [169]
C4.5 (5xCV)	77.8%		Bennett & Blue [8]
IB 1c	77.6%		KIS (WEKA)
QDA	75.4%		Ster & Dobnikar [169]
LFC	75.1%		Ster & Dobnikar [169]
ASI	74.4%		Ster & Dobnikar [169]
OC1 (5xCV)	71.7%		Bennett & Blue [8]
1R	71.0%		KIS (WEKA)
FOIL	66.4%		KIS (WEKA)
C4.5-reguły	53.8%	5.9	Zarndt [188]
IB 1-4	46.2%		KIS (WEKA)

Tabela 3.4: Wyniki 10-krotnej kroswalidacji dla danych *Cleveland heart disease*

dość zrozumiała nawet dla laika, bo mówi o wysokim stopniu złośliwości (degmalig) oraz dużej liczbie zaatakowanych chorobą węzłów (inv-nodes). Niestety niewiele więcej ciekawych informacji można wydobyć z tego zestawu danych. Testy kroswalidacyjne nie dają tak wysokiej poprawności jak ta prosta reguła, dla żadnego systemu klasyfikacji. Także samo drzewo SSV przy drobnych zaburzeniach danych często generuje inne zestawy reguł, na przykład:

$\mathcal{R}_1$: breast = left $\wedge$ inv-nodes $> 2.5 \rightarrow$ *recurrence-events*
$\mathcal{R}_2$: *else no-recurrence-events*

Dokładność tej reguły na całości danych, to 75.5% (wrażliwość 30.5%, znamienność 94.5%). Przesłanka „breast = left" pozwala przypuszczać, że jest to jednak reguła bez wartości, oraz, że dane są bardzo zaszumione i nie niosą w sobie wystarczającej do rozwiązania problemu diagnozowania ilości informacji. Wyniki testu 10-krotnej kroswalidacji przedstawia tabela 3.5.

Metoda	Poprawność	Odch.st.	Źródło
MML	75.3%	7.8	Zarndt [188]
C4.5-drzewo	73.9%	7.1	Zarndt [188]
MLP BP	73.5%	9.4	Zarndt [188]
SSV	72.7%	1.4	
IB 1	71.8%	7.5	Zarndt [188]
CART	71.4%	5.0	Zarndt [188]
CN2	70.7%	7.8	Zarndt [188]
C4.5-reguły	69.7%	7.2	Zarndt [188]
Naive Bayes	69.3%	10.0	Zarndt [188]
Weighted networks	68-73.5%		Tan & Eshelman [173]
IB 3	67.9%	7.7	Zarndt [188]
ID3	66.2%	8.5	Zarndt [188]
AQ15	66-72%		Statlog [129]

Tabela 3.5: Wyniki 10-krotnej kroswalidacji dla danych *Ljubljana breast cancer*

Konsekwencją trudności problemu są zwykle wysokie wartości odchyleń standardowych wyników. W efekcie prawie wszystkie różnice widoczne w tabeli 3.5 należy uznać za statystycznie nieistotne.

Mushroom

Zbiór *Mushroom* zawiera 8124 wektory z przestrzeni 22 cech symbolicznych, które łącznie mogą przyjmować 125 wartości. Dane podzielone są na dwie klasy (4208 grzybów jadalnych i 3916 trujących).

Drzewo decyzji oparte na kryterium SSV jest jednym z wielu systemów, które klasyfikują ten zbiór z dokładnością 100% (również w testach kroswalidacyjnych).

Opis danych wygenerowany przez SSV jest i w tym przypadku jednym z najprostszych znanych z literatury:

$\mathcal{R}_1$: odor $\notin$ {a,l,n} $\rightarrow$ *poisonous*
$\mathcal{R}_2$: spore-print-color $\in$ {r,w} $\wedge$ population = v $\wedge$ habitat $\notin$ {l,p} $\rightarrow$ *poisonous*
$\mathcal{R}_3$: spore-print-color $\in$ {r,w} $\wedge$ population $\neq$ v $\wedge$ gill-size $\neq$ b $\rightarrow$ *poisonous*
$\mathcal{R}_4$: *else edible*

Pierwsza reguła oddziela 3796 grzybów trujących pozostawiając do dalszego opisu 120. Oznacza to, że najważniejszą cechą pozwalającą rozpoznawać grzyby jadalne od trujących jest zapach.

Ponieważ zadanie klasyfikacji grzybów daje się bardzo dobrze rozwiązać przez stopniowe oddzielanie grzybów trujących od jadalnych, wyjątkowo dobre wyniki można uzyskać tworząc drzewo klasyfikacji poprzez hierarchiczny wybór podziałów maksymalizujących poprawność klasyfikacji zbioru treningowego. Algorytm szukania wiązką pozwala wówczas uzyskać następujący zestaw reguł:

$\mathcal{R}_1$: odor $\notin$ {a,l,n} $\rightarrow$ *poisonous*
$\mathcal{R}_2$: spore-print-color = r $\rightarrow$ *poisonous*
$\mathcal{R}_3$: gill-size = n $\wedge$ stalk-surface-above-ring $\in$ {k,y} $\rightarrow$ *poisonous*
$\mathcal{R}_4$: gill-size = n $\wedge$ population = c $\rightarrow$ *poisonous*
$\mathcal{R}_5$: *else edible*

Kolejne reguły, dodawane do zestawu, redukują błąd klasyfikacji odpowiednio do 120, 48, 8 i 0. Tak duża skuteczność tego prostego algorytmu jest jednak w tym przypadku wyjątkiem – dla innych problemów wyniki są zdecydowanie mniej atrakcyjne.

Voting

Problem klasyfikacji o nazwie *Voting* dotyczy rozpoznawania przynależności partyjnej 435 członków Kongresu Stanów Zjednoczonych (267 demokratów i 168 republikanów) na podstawie głosów oddanych przez nich w 16 głosowaniach.

Bardzo prosta reguła:

$\mathcal{R}_1$: physician-fee-freeze = y $\rightarrow$ *republicans*
$\mathcal{R}_2$: *else democrats*

klasyfikuje zbiór z dokładnością 95.6%. Nieco bardziej złożony zestaw reguł, znaleziony za pomocą drzewa SSV, klasyfikuje poprawnie 96.3% wektorów:

$\mathcal{R}_1$: physician-fee-freeze = y $\wedge$ synfuels-corporation-cutback $\neq$ y $\rightarrow$ *republicans*

$\mathcal{R}_2$: physician-fee-freeze = y $\wedge$ synfuels-corporation-cutback = y $\wedge$ mx-missile = n $\rightarrow$ *republicans*

$\mathcal{R}_3$: *else democrats*

Wyniki 10-krotnej kroswalidacji przedstawia tabela 3.6. Zakładając, że odchy-

Metoda	Poprawność	Odch.st.	Źródło
IND mml opt	96.33%		Lim et al. [119, 118]
IND bayes opt	96.14%		Lim et al. [119, 118]
S-Plus 1-SE	95.68%		Lim et al. [119, 118]
1R	95.65%		Lim et al. [119, 118]
FACT	95.65%		Lim et al. [119, 118]
IndCART 1-SE	95.65%		Lim et al. [119, 118]
OC1	95.65%		Lim et al. [119, 118]
QUEST 0-SE	95.65%		Lim et al. [119, 118]
QUEST 1-SE	95.65%		Lim et al. [119, 118]
SSV	95.57%	0.1	
Cal5	95.43%		Lim et al. [119, 118]
QUEST komb.lin. 1-SE	95.43%		Lim et al. [119, 118]
FACT komb.lin.	95.43%		Lim et al. [119, 118]
IndCART 0-SE	95.20%		Lim et al. [119, 118]
C4.5-drzewo	95.20%		Lim et al. [119, 118]
LMDT	95.17%		Lim et al. [119, 118]
S-Plus 0-SE	95.00%		Lim et al. [119, 118]
QUEST komb.lin. 0-SE	94.97%		Lim et al. [119, 118]
IND mml	94.97%		Lim et al. [119, 118]
IND bayes	94.74%		Lim et al. [119, 118]
C4.5-reguły	94.74%		Lim et al. [119, 118]
OC1 komb.lin.	94.26%		Lim et al. [119, 118]
OC1 mix	94.20%		Lim et al. [119, 118]

Tabela 3.6: Wyniki 10-krotnej kroswalidacji dla danych *Voting*

lenia standardowe różnic będą bliskie odchyleniu dla metody SSV, można wyciągnąć wniosek, że większość wyników zajmujących miejsca powyżej SSV mieści się w granicach statystycznej nieistotności różnic. Tylko dwie metody zajmujące czołowe miejsca w tabeli dają istotnie wyższe dokładności klasyfikacji.

Bardzo dobry wynik metody 1R wskazuje, że dla tych danych drzewa aby dobrze generalizować muszą być bardzo mocno obcinane. To, że niektóre algorytmy drzew uzyskują gorsze wyniki niż 1R może świadczyć o tym, że strategie doboru sposobów obcinania nie zawsze dają optymalne wyniki.

Australian Credit

Na dane *Australian Credit* składa się 690 wektorów należących do dwóch klas i opisanych 14 cechami (6 ciągłymi i 8 symbolicznymi).

Reguły uzyskane z drzewa SSV klasyfikują dane z dokładnością 87.2%:

$\mathcal{R}_1$: $F_8 = t \wedge F_9 = t \rightarrow$ *class +*
$\mathcal{R}_2$: $F_8 = t \wedge F_9 = f \wedge F_5 \notin \{ff, d, i, aa, m, c\} \rightarrow$ *class +*
$\mathcal{R}_3$: *else class -*

Niestety z braku opisów cech i ich wartości (nie udostępnionych z powodu tajności danych) nie można przełożyć tych prostych reguł na zrozumiałe terminy.

Zestawienie wyników 10-krotnej kroswalidacji dla tych danych przedstawia tabela 3.7. Uwzględniając odchylenie standardowe uzyskane dla modelu SSV, bez

Metoda	Poprawność	Odch.st.	Źródło
Cal5	86.9%		Statlog [130]
ITRULE	86.3%		Statlog [130]
LogDA	85.9%		Statlog [130]
DIPOL92	85.9%		Statlog [130]
LDA	85.9%		Statlog [130]
SSV	85.8%	0.7	
CART	85.5%		Statlog [130]
RBF	85.5%		Statlog [130]
CASTLE	85.2%		Statlog [130]
Naive Bayes	84.9%		Statlog [130]
IndCART	84.8%		Statlog [130]
MML	84.8%	4.8	Zarndt [188]
MLP BP	84.6%		Statlog [130]
C4.5	84.5%		Statlog [130]
SMART	84.2%		Statlog [130]
Bayes Tree	82.9%		Statlog [130]
kNN	81.9%		Statlog [130]
NewID	81.9%		Statlog [130]
AC^2	81.9%		Statlog [130]
LVQ	80.3%		Statlog [130]
ALLOC80	79.9%		Statlog [130]
CN2	79.6%		Statlog [130]
QDA	79.3%		Statlog [130]

Tabela 3.7: Wyniki 10-krotnej kroswalidacji dla danych *Australian credit*

testów sprawdzających różnice pomiędzy dokładnościami klasyfikacji różnych

systemów przy tych samych losowaniach kroswalidacyjnych, nie można twierdzić o statystycznej istotności różnic wyników z początku tabeli.

Image segmentation

Zbiór *Image segmentation* liczy 2310 wektorów. Każdy ma 19 współrzędnych rzeczywistych, które określają pewne własności 9-punktowych (3×3) obszarów wybranych losowo z 7 plenerowych zdjęć. Próbki zostały ręcznie sklasyfikowane, w zależności od tego co przedstawiały, do 7 klas (takich jak: trawa, niebo, cegły itp.).

Wyniki 10-krotnej kroswalidacji przedstawia tabela 3.8. Ponieważ nie znamy

Metoda	Poprawność	Odch.st.	Źródło
ALLOC80	97.0%		Statlog [130]
AC^2	96.9%		Statlog [130]
Bayes Tree	96.7%		Statlog [130]
NewID	96.6%		Statlog [130]
DIPOL92	96.1%		Statlog [130]
CART	96.0%		Statlog [130]
C4.5	96.0%		Statlog [130]
SSV	95.9%	0.3	
CN2	95.7%		Statlog [130]
IndCART	95.5%		Statlog [130]
QUEST 0-SE	95.4%		Lim et al. [119, 118]
LVQ	95.4%		Statlog [130]
QUEST 1-SE	95.1%		Lim et al. [119, 118]
SMART	94.8%		Statlog [130]
MLP BP	94.6%		Statlog [130]
Cal5	93.8%		Statlog [130]
Kohonen	93.3%		Statlog [130]
RBF	93.1%		Statlog [130]
kNN	92.3%		Statlog [130]
LDA	91.7%		Lim et al. [119, 118]
LogDA	89.1%		Statlog [130]
CASTLE	88.8%		Statlog [130]
LDA	88.4%		Statlog [130]
QDA	84.3%		Statlog [130]
Naive Bayes	73.5%		Statlog [130]
ITRULE	54.5%		Statlog [130]

Tabela 3.8: Wyniki 10-krotnej kroswalidacji dla danych *Image segmentation*

odchyleń standardowych pozostałych wyników, trudno wyrokować o statystycznej istotności ich różnic. Zakładając, że wszystkie metody z górnej części tabeli mają to samo odchylenie (0.3), różnica poprawności pozwalająca z prawdopodobieństwem 0.95 twierdzić o uzyskiwaniu lepszych wyników niż SSV wynosi $1.812 \cdot \sqrt{0.3^2 + 0.3^2} \simeq 0.76$. Zatem przy takim założeniu, metody zajmujące trzy czołowe pozycje (bądź dwie – w zależności od sposobu zaokrąglania) można uznać za istotnie lepsze w klasyfikacji tego zbioru niż SSV.

NASA Shuttle

Dane *NASA Shuttle* udostępnione zostały z podziałem na część treningową i testową. Wektory opisane są 9 cechami ciągłymi i reprezentują 7 różnych klas. Zbiór treningowy zawiera 43500 obiektów a testowy 14500. Klasa 1 zdecydowanie dominuje w zbiorze treningowym (78.4%) i testowym (79.1%). 10-krotne trenowanie drzewa decyzji SSV kończyło się bez obcięcia jakichkolwiek liści drzewa bądź z obcięciem jednego rozgałęzienia, które skutkowało jednym błędnie klasyfikowanym wektorem zbioru treningowego. Niezależnie od tego, powstałe reguły popełniały 1 błąd na zbiorze testowym. Pełny zestaw reguł opisujący drzewo (po usunięciu zbędnych przesłanek) wygląda następująco:

$\mathcal{R}_1$: $F_1 < 54.5 \wedge F_9 < 3 \wedge F_2 > -26.5 \rightarrow$ *class 1*
$\mathcal{R}_2$: $F_9 > 3 \wedge F_1 < 39.5 \wedge F_2 > -30.5 \rightarrow$ *class 1*
$\mathcal{R}_3$: $F_1 < 54.5 \wedge F_2 < -26.5 \wedge F_9 < 1 \rightarrow$ *class 1*
$\mathcal{R}_4$: $F_1 > 54.5 \wedge F_5 > 53 \wedge F_1 < 68.5 \wedge F_2 > -16 \rightarrow$ *class 1*

$\mathcal{R}_5$: $F_9 > 3 \wedge F_1 > 39.5 \wedge F_1 < 52.5 \wedge F_2 > -2 \rightarrow$ *class 2*
$\mathcal{R}_6$: $F_5 < 53 \wedge F_1 > 57.5 \wedge F_7 > 6.5 \wedge F_2 > -3.5 \wedge F_2 < 406.5 \rightarrow$ *class 2*

$\mathcal{R}_7$: $F_9 > 3 \wedge F_1 < 39.5 \wedge F_2 < -30.5 \wedge F_2 > -547.5 \rightarrow$ *class 3*
$\mathcal{R}_8$: $F_9 > 3 \wedge F_1 > 39.5 \wedge F_1 < 52.5 \wedge F_2 < -2 \rightarrow$ *class 3*
$\mathcal{R}_9$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 > 20.5 \wedge F_2 < -41 \rightarrow$ *class 3*
$\mathcal{R}_{10}$: $F_1 < 54.5 \wedge F_9 < 3 \wedge F_2 < -26.5 \wedge F_9 > 1 \wedge F_2 > -721 \rightarrow$ *class 3*
$\mathcal{R}_{11}$: $F_5 < 53 \wedge F_1 > 57.5 \wedge F_7 > 6.5 \wedge F_2 < -3.5 \rightarrow$ *class 3*
$\mathcal{R}_{12}$: $F_5 > 53 \wedge F_1 > 68.5 \wedge F_2 < -21.5 \rightarrow$ *class 3*
$\mathcal{R}_{13}$: $F_1 > 54.5 \wedge F_5 > 53 \wedge F_1 < 68.5 \wedge F_2 < -16 \rightarrow$ *class 3*
$\mathcal{R}_{14}$: $F_1 < 54.5 \wedge F_9 > 3 \wedge F_1 > 52.5 \wedge F_2 < -47 \rightarrow$ *class 3*

$\mathcal{R}_{15}$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 < 20.5 \wedge F_8 > 63 \rightarrow$ *class 4*
$\mathcal{R}_{16}$: $F_1 < 54.5 \wedge F_9 > 3 \wedge F_1 > 52.5 \wedge F_2 > -47 \rightarrow$ *class 4*
$\mathcal{R}_{17}$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 > 20.5 \wedge F_2 > -41 \wedge F_2 < 859.5 \rightarrow$ *class 4*

$\mathcal{R}_{18}$: $F_5 < 53 \wedge F_1 > 57.5 \wedge F_7 < 6.5 \wedge F_9 > 46 \rightarrow$ *class 5*
$\mathcal{R}_{19}$: $F_1 > 68.5 \wedge F_2 > -21.5 \rightarrow$ *class 5*
$\mathcal{R}_{20}$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 < 20.5 \wedge F_8 < 63 \wedge F_2 < 514.5 \rightarrow$ *class 5*

$\mathcal{R}_{21}$: $F_5 < 53 \wedge F_1 > 57.5 \wedge F_7 > 6.5 \wedge F_2 > 406.5 \rightarrow$ *class 6*
$\mathcal{R}_{22}$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 < 20.5 \wedge F_8 < 63 \wedge F_2 > 514.5 \rightarrow$ *class 6*
$\mathcal{R}_{23}$: $F_5 < 53 \wedge F_1 > 57.5 \wedge F_7 < 6.5 \wedge F_9 < 46 \rightarrow$ *class 6*
$\mathcal{R}_{24}$: $F_1 > 54.5 \wedge F_5 < 53 \wedge F_1 < 57.5 \wedge F_7 > 20.5 \wedge F_2 > 859.5 \rightarrow$ *class 6*

$\mathcal{R}_{25}$: $F_9 > 3 \wedge F_1 < 39.5 \wedge F_2 < -547.5 \rightarrow$ *class 7*
$\mathcal{R}_{26}$: $F_1 < 54.5 \wedge F_9 < 3 \wedge F_9 > 1 \wedge F_2 < -721 \rightarrow$ *class 7*

Ponieważ reguły te łącznie wypełniają całą przestrzeń, dla dalszego uproszczenia tego zestawu reguł można usunąć reguły dla klasy 3 (jako najliczniejsze) i zastąpić je klauzulą *w przeciwnym przypadku* – w ten sposób uzyskalibyśmy zestaw 18 reguł (+ warunek *else*).

Tabela 3.9 pokazuje wyniki na testowej części danych uzyskiwane przez różne systemy.

Metoda	Poprawność	Źródło
SSV	99.99%	
NewID	99.99%	Statlog [130]
Bayes Tree	99.98%	Statlog [130]
CN2	99.97%	Statlog [130]
Cal5	99.97%	Statlog [130]
CART	99.92%	Statlog [130]
IndCART	99.91%	Statlog [130]
C4.5	99.90%	Statlog [130]
AC^2	99.68%	Statlog [130]
ITRULE	99.59%	Statlog [130]
MLP BP	99.57%	Statlog [130]
LVQ	99.56%	Statlog [130]
kNN	99.56%	Statlog [130]
DIPOL92	99.52%	Statlog [130]
SMART	99.41%	Statlog [130]
ALLOC80	99.17%	Statlog [130]
RBF	98.60%	Statlog [130]
CASTLE	96.20%	Statlog [130]
LogDA	96.17%	Statlog [130]
Naive Bayes	95.50%	Statlog [130]
LDA	95.17%	Statlog [130]
QDA	93.28%	Statlog [130]
MML	78.6%	Zarndt [188]

Tabela 3.9: Wyniki testu dla danych *NASA Shuttle*

Mimo dużego rozmiaru danych system drzew SSV, uczący się przez 10-krotną kroswalidację, produkuje wynik w krótkim czasie (dla procesora Pentium III z zegarem 733 MHz ok. 30 s, włączając w to nakład potrzebny do obsługi środowiska graficznego). Jest to (tak samo jak w przypadku danych *Hypothyroid*) potwierdzenie faktu, że przy małej złożoności problemu, drzewa decyzji mogą działać bardzo szybko nawet dla bardzo dużych zbiorów danych.

Pima Indians diabetes

Baza danych *Pima Indians diabetes* to 768 wektorów podzielonych na dwie grupy: zdrowych (*healthy* – 500 obiektów) oraz chorych (*diabetes* – 268 obiektów). Przestrzeń klasyfikacji ma wymiar 8 i wszystkie jej cechy są ciągłe.

Metoda C-MLP2LN znalazła reguły opisujące zbiór z dokładnością 75% [45]:

$\mathcal{R}_1$: $F_2 \leqslant 151 \wedge F6 \leqslant 47 \rightarrow$ *healthy*
$\mathcal{R}_2$: *else diabetes*

Drzewo SSV znajduje prostszą regułę o takiej samej dokładności:

$\mathcal{R}_1$: $F_2 < 143.5 \rightarrow$ *healthy*
$\mathcal{R}_2$: *else diabetes*

Porównanie wyników 12-krotnej kroswalidacji przedstawia tabela 3.10. Spośród systemów generujących reguły tylko QUEST, ITRULE oraz Cal5 osiągnęły lepsze wyniki, aczkolwiek, w związku z dość dużą wariancją różnice między nimi nie są statystycznie istotne.

Melanoma (czerniak)

Dane na temat raka skóry zostały zebrane w Centrum Dermatologii w Rzeszowie [6]. Każdy z 250 przypadków zbioru treningowego oraz 26 testowego, jest opisany 14 cechami. 13 z nich określa pewne zewnętrzne cechy badanego znamienia. Dodatkowo podana jest wartość współczynnika TDS (Total Dermatoscopy Score), który jest kombinacją liniową pozostałych 13 wartości.

Do tego zadania klasyfikacji zastosowane zostały systemy: GTS (*General-to-Specific*) [89, 88], LERS (*Learning from Examples based on Rough Sets*) [80, 81], a także FSM (*Feature Space Mapping*) [1], SBM (*Similarity Based Methods*) [79] oraz SSV i MLP2LN [52]. Uzyskane poprawności na zbiorach treningowym i testowym przedstawia tabela 3.11. Uproszczenia GTS są możliwe tylko przy interakcji użytkownika, stąd test mógł być wykonany tylko raz. W przypadkach, w których możliwe było 10-krotne powtórzenie testu, po znaku $\pm$ podane są odchylenia standardowe poprawności. Dla metody *SBM* w kolumnie *Liczba reguł* podane są liczby wektorów referencyjnych (kandydatów na najbliższych sąsiadów).

Algorytm budowania drzew w oparciu o kryterium SSV jest dla tych danych bardzo stabilny: także w 10-krotnej kroswalidacji (na zbiorze treningowym lub na sumie obu zbiorów), w każdym przebiegu, generuje ten sam zestaw reguł:

$\mathcal{R}_1$: TDS $< 4.85 \wedge$ C.BLUE = absent $\rightarrow$ *Benign nevus*
$\mathcal{R}_2$: TDS $< 4.85 \wedge$ C.BLUE = present $\rightarrow$ *Blue nevus*

Metoda	Poprawność	Odch.st.	Źródło
LogDA	77.7%		Statlog [130]
DIPOL92	77.6%		Statlog [130]
LDA	77.5%		Statlog [130]
SMART	76.8%		Statlog [130]
QUEST 0-SE	76.7%		Lim et al. [119, 118]
QUEST 1-SE	75.7%		Lim et al. [119, 118]
RBF	75.7%		Statlog [130]
ITRULE	75.5%		Statlog [130]
MML (10xCV)	75.5%	6.3	Zarndt [188]
MLP BP	75.2%		Statlog [130]
Cal5	75.0%		Statlog [130]
SSV	74.8%	1.2	
CART (10xCV)	74.7%	5.4	Zarndt [188]
CART	74.5%		Statlog [130]
CASTLE	74.2%		Statlog [130]
Naive Bayes	73.8%		Statlog [130]
QDA	73.8%		Statlog [130]
C4.5	73.0%		Statlog [130]
IndCART	72.9%		Statlog [130]
Bayes Tree	72.9%		Statlog [130]
LVQ	72.8%		Statlog [130]
Kohonen	72.7%		Statlog [130]
C4.5-drzewo (10xCV)	72.7%	6.6	Zarndt [188]
AC^2	72.4%		Statlog [130]
NewID	71.1%		Statlog [130]
CN2	71.1%		Statlog [130]
ALLOC80	69.9%		Statlog [130]
kNN	67.6%		Statlog [130]
C4.5-reguły (10xCV)	67.0%	2.9	Zarndt [188]

Tabela 3.10: Wyniki 12-krotnej kroswalidacji dla danych *Pima Indians diabetes*

Metoda	Liczba reguł	Popr.tren.	Popr.test.
SSV	4	$100\% \pm 0.0$	$100\% \pm 0.0$
MLP2LN	4	98.0%	100%
GTS uproszczony	4	97.6%	100%
SBM 2 cechy	13	97.5%	100%
SBM 2 cechy	250	$97.4\% \pm 0.3$	$100\% \pm 0.0$
FSM f.prostokątne	7	$95.5\% \pm 1.0$	$100\% \pm 0.0$
LERS	21	—	96.2%
kNN k=1	250	—	96.2
FSM f.gaussowskie	15	$93.7\% \pm 1.0$	$95\% \pm 3.6$
GTS podstawowy	198	85%	84.6%

Tabela 3.11: Wyniki treningowe i testowe dla danych *Melanoma*

$\mathcal{R}_3$: TDS $\in (4.85, 5.45) \rightarrow$ *Suspicious*
$\mathcal{R}_4$: TDS $> 5.45 \rightarrow$ *Malignant*

i osiąga 100% poprawności na częściach treningowych i testowych.

Zawody NIPS 2000

W ramach współzawodnictwa rozgrywanego pod nazwą *Unlabeled Data Supervised Learning Competition* przygotowywane są i udostępniane chętnym do podjęcia rywalizacji problemy klasyfikacji oraz regresji. Każde z zadań zawiera trzy zbiory: dwa treningowe i jeden testowy. Tylko jeden ze zbiorów treningowych ma przypisane wektorom etykiety klas. Drugi zbiór treningowy jest zwykle zdecydowanie większy od pierwszego, ale nie ma przydzielonych etykiet (stąd nazwa konkursu). Zbiór testowy udostępniany jest również bez etykiet, choć są one odpowiednio przydzielone i dostępne dla jurorów konkursu. Każdy, kto chce wziąć udział w rywalizacji może pobrać (za pośrednictwem internetu) dla każdego problemu wspomniane trzy zbiory i tworzyć modele dla tych danych, by później odesłać zbiór testowy z przydzielonymi etykietami. Zbiór ten jest automatycznie sprawdzany, a uzyskana poprawność pokazywana jako wynik transakcji.

System klasyfikacji SSV wziął udział w konkursie zorganizowanym w ramach konferencji NIPS (*Neural Information Processing Systems* w 2000 roku [116]. Zadania były udostępniane w czterech rundach. W sumie przedstawiono 7 problemów klasyfikacji oraz 4 problemy regresji.

W jednym z problemów klasyfikacji drzewo SSV wykryło cechę, która była doskonale skorelowana z etykietą klasy. Efektem była niemal 100% dokładność również na zbiorze testowym. Zapewne wiele różnych systemów dostrzegło tę zależność i problem ten wycofano z konkursu.

Drzewa SSV wzięły udział w rywalizacji dotyczącej 5 problemów klasyfikacji. W jednym z problemów drzewo SSV okazało się najskuteczniejsze (taki sam wynik uzyskała również metoda oparta na SVM użyta przez Thorsten'a Joachims'a).

Za pierwsze miejsce przyznawano 5 punktów, za drugie 4, trzecie – 3, czwarte – 2, piąte – 1. W klasyfikacji łącznej system SSV zajął piąte miejsce na 31 sklasyfikowanych (jeden z systemów, które zajęły ex aequo pierwsze miejsce, był reprezentowany przez osobę współpracującą z organizatorami konkursu i nie brał udziału w naliczaniu punktów). Zestawienie systemów, które zdobyły punkty w konkursie przedstawia tabela 3.12. Dla każdego z systemów kolumna *próby* mówi

punkty	system	próby	pr.sk.	podium	pr.sk./podium	punkty/pr.sk.
15	MPLs	416	416	6	69.33	0.04
14	MISC	86	72	5	14.40	0.19
13	Stoch.Discr.	29	10	3	3.33	1.30
10	Bez nazwy	26	20	2	10.00	0.50
6	SSV	10	3	2	1.50	2.00
5	ILLM	2	1	1	1.00	5.00
5	TSVM	1	1	1	1.00	5.00
4	CROMRSel	52	52	1	52.00	0.08
4	Dec.Tree	2	2	1	2.00	2.00
4	MultiDT	111	85	1	85.00	0.05
3	VRM	5	2	1	2.00	1.50
3	MLP	16	2	1	2.00	1.50
2	HONG	6	1	1	1.00	2.00
1	FSM	4	1	1	1.00	1.00
1	MET	3	1	1	1.00	1.00
1	ILLM SG	1	1	1	1.00	1.00

Tabela 3.12: Wyniki konkursu UDSLC NIPS 2000

ile razy łącznie przesyłano wyniki dla wszystkich sześciu problemów, kolumna *pr.sk.* (*próby skuteczne*) podaje podobną sumę z ograniczeniem do tych problemów, w których uzyskano punkty, kolumna *podium* mówi ile razy wynik był w pierwszej piątce, a pozostałe dwie kolumny to odpowiednie proporcje.

Jak widać z tabeli i w tym przypadku wyniki porównania systemów nie są całkiem wiarygodne. W tak zorganizowanym konkursie również jest miejsce do nadużyć – wielokrotne przesyłanie wyników jest metodą na pobieranie informacji o zbiorze testowym, więc najlepiej byłoby, gdyby dla każdego systemu, można było tylko jednokrotnie przesłać rozwiązania znalezione dla danego problemu. Ocena klasyfikatorów uwzględniająca dwie ostatnie kolumny tabeli, byłaby z pewnością jeszcze korzystniejsza dla drzew SSV.

	Wyniki SSV: lepsze/takie same/gorsze		
	Testy jednokrotne	CV – bezwzględnie	CV – stat. znacząco
C4.5	2/0/0	5/0/2	2/5/0
Cal5	2/0/0	2/0/2	1/3/0
CART	3/0/0	6/0/1	1/6/0
QUEST	1/0/0	2/0/2	1/3/0

Tabela 3.13: Porównanie wyników SSV i innych systemów (trójki liczb mówią ile razy drzewa SSV uzyskały wyniki lepsze/takie same/gorsze niż odpowiedni system)

Podsumowanie wyników

Drzewa decyzji budowane z wykorzystaniem kryterium separowalności SSV wykazały wysoką jakość klasyfikacji w testach generalizacji. W każdym z przedstawionych przypadków otrzymano dodatkowo zwarty opis problemu klasyfikacji w języku logiki klasycznej pierwszego rzędu.

Parametry definiujące sposób generowania drzew SSV takie jak metoda obcinania (do stopnia czy do liczby liści) oraz liczba przebiegów kroswalidacji wykonywanej wewnętrznie w procesie uczenia, zwykle wpływają na ostateczny wynik w małym (zaniedbywalnym) stopniu. Dokładniejsza analiza wpływu tych parametrów na uzyskiwane wyniki oraz opracowanie metod ich automatycznego doboru, prowadzą do zagadnienia uczenia na poziomie *meta* i mogą być przedmiotem dalszych badań (p. rozdział 3.11).

Rzetelne porównanie otrzymanych wyników z rezultatami innych systemów jest zadaniem bardzo trudnym ze względu na różnorodność podejść, w wyniku których uzyskano rezultaty dostępne w literaturze. Prace prezentujące wyniki, czasami pomijają pewne istotne szczegóły.

Jedną z prób porównania wyników może być ocena (na podstawie zebranych w tabelkach wyników) ile razy dana metoda była lepsza od innej z uwzględnieniem, bądź nie, statystycznej istotności różnic. Tabela 3.13 prezentuje takie zestawienie, porównujące wyniki uzyskane metodą drzew SSV oraz najpopularniejszymi metodami drzew decyzji.

Dla danej metody pokazane są liczby rozpatrywanych zadań klasyfikacji, w których drzewa SSV uzyskały lepsze, jednakowe i gorsze rezultaty. Pierwsza kolumna liczb przedstawia porównanie wyników dla danych, na których wykonywane były jednokrotne testy, tj. danych z wydzielonym zbiorem testowym (*Hypothyroid* i *Shuttle*) oraz danych, dla których wykonywany był test *leave-one-out* (*Appendicitis*). Kolumny druga i trzecia porównują wyniki testów kroswalidacji (druga porównuje średnie wyniki bezwzględnie, a trzecia z uwzględnieniem sta-

tystycznej istotności różnic).

Dla wielu średnich poprawności klasyfikacji, zaczerpniętych z literatury, nie znamy ich odchyleń standardowych. W takich sytuacjach ocena statystycznej istotności różnic wyników (testem t Studenta z progiem istotności 0.95 – p. rozdział 2.2.3) została wykonana z założeniem że odchylenia te są takie same jak dla systemu SSV. Odchylenia standardowe podawane przez Zarndta [188] są dużo wyższe niż dostępne z innych źródeł (także niż te wyliczone dla SSV), co pozwala sądzić, że są to odchylenia wewnętrznych wyników kroswalidacji, a nie średnich tych wyników. Dlatego też, dla oszacowania odchylenia standardowego średnich wyników, zostały one podzielone przez $\sqrt{n}$, gdzie n jest liczbą przebiegów kroswalidacji.

W przypadku, kiedy dla danego systemu dostępnych było kilka różnych wyników (np. dla CART i QUEST wyniki dla metod przycinania 0-SE i 1-SE), do oszacowania statystycznej istotności różnic zostały użyte ich wartości średnie.

Zestawienie wyników przedstawione w tabeli 3.13 potwierdza bardzo wysoką skuteczność systemu drzew decyzji SSV. We wszystkich testach wykonywanych jednokrotnie drzewa SSV uzyskały lepsze wyniki niż inne metody indukcji drzew (różnice są często bardzo małe, jednak charakter testów nie pozwala zakładać rozkładu normalnego wyników, a więc nie ma sensu badać statystycznej istotności różnic testem t Studenta – p. rozdział 2.2.3).

W testach kroswalidacyjnych, choć kilkakrotnie system SSV uzyskał wyniki niższe niż inne systemy, to ani razu różnice te nie były statystycznie znaczące. Zdarzało się, natomiast (przynajmniej raz dla każdej z alternatywnych metod), że wynik systemu SSV był istotnie wyższy.

3.3 Drzewa heterogeniczne

Drzewa decyzji, które się spotyka w zastosowaniach, najczęściej opierają się wyłącznie na decyzjach dotyczących pojedynczych cech, co sprawia, że granice decyzji takich klasyfikatorów są zawsze prostopadłe do osi definiujących przestrzeń klasyfikacji. Nie jest to jednak ograniczenie podejścia wykorzystującego drzewa decyzji, bo łatwo można skonstruować drzewa, które będą zawierały przesłanki całkiem innego typu. W literaturze można spotkać rozwiązania, w których warunki podziału węzłów dotyczą kombinacji liniowych cech [16]. Ten sam efekt ukośnych granic decyzji można uzyskać korzystając na poziomie każdego węzła z metod liniowej dyskryminacji [24, 37]. Są też takie podejścia, w których każdy węzeł drzewa jest generowany przy użyciu osobnego systemu klasyfikacji (np. modelu SVM [8]). Buduje się również drzewa sieci neuronowych (w rzeczywistości są to duże sieci o hierarchicznej strukturze) [63], natomiast nie widać w literaturze takich drzew klasyfikacji, które do wyznaczania przynależności do wę-

złów wykorzystują odległości w przestrzeni.

Nic nie stoi na przeszkodzie, aby dowolny system drzew decyzji mógł wykorzystać w podziałach miary odległości, które znacznie wzbogacają systemy tworzenia drzew dodając możliwość generowania różnorodnych granic decyzji. Jeśli do oryginalnej przestrzeni klasyfikacji dodamy wymiar, który będzie określał odległość danego wektora od pewnego punktu w przestrzeni, to każdy algorytm drzew decyzji będzie mógł wykorzystać ten wymiar tak jak każdy inny wymiar przestrzeni, jednak prostopadła do jego osi granica decyzji może mieć najróżniejsze kształty w oryginalnej przestrzeni klasyfikacji (podobne własności wykorzystuje się w modelach SVM). Rozwiązanie to jest bardzo ogólne – można w ten sposób tworzyć przesłanki realizujące dowolne funkcje (kombinacje liniowe, funkcje wyższych stopni itd.). Otrzymujemy więc w ten sposób heterogeniczne drzewa, które mają większe możliwości dopasowania się do różnych danych treningowych, ale oczywiście proces ich szukania istotnie się wydłuża. Liczba możliwych do stworzenia cech jest oczywiście nieskończona, należy się więc w praktyce ograniczyć do pewnej niezbyt licznej rodziny funkcji. Użycie w

Praktyczna realizacja heterogenicznych drzew SSV, oprócz oryginalnych cech przestrzeni klasyfikacji, analizuje odległości (mierzone metryką euklidesową) od wektorów znajdujących się w zbiorze danych treningowych. Dla dodatkowego ograniczenia złożoności problemu można w każdym z węzłów rozpatrywać jedynie odległości od wektorów wpadających do tego węzła. Ograniczenie to ma sens wtedy, kiedy mamy bardzo dużo danych treningowych i nie można trzymać w pamięci roboczej pełnej macierzy odległości. Inaczej macierz taką można wyliczyć raz na początku, ograniczając w ten sposób nakład czasu potrzebny na wyliczenie odległości w każdym z węzłów.

Dzięki stosunkowo małej złożoności algorytmów drzew decyzji (a więc i drzew SSV) w efekcie uzyskujemy systemy klasyfikacji, których złożoność jest akceptowalna dla realnych zastosowań.

Choć w oryginalnym ujęciu drzewo SSV jest całkowicie niezależne od skalowania poszczególnych cech, to jednak w przypadku użycia przesłanek testujących odległości należy mieć na względzie, że faza przygotowań danych (standaryzacja itp.) może mieć kluczowy wpływ na wyniki systemu. Podobnie, duże znaczenie dla uzyskiwanych wyników może mieć dobór metryki, selekcja cech wyznaczających przestrzeń, w której liczone są odległości czy też przypisywanie wag poszczególnym cechom. Oczywiście stosując takie rozszerzenia należy bardzo uważać, by nie spowodowało to nadmiernego wzrostu złożoności obliczeniowej metody.

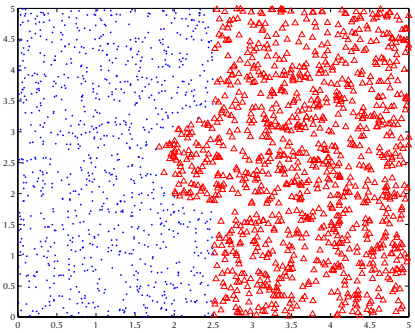
Oznaczenie : W prezentowanych poniżej regułowych opisach heterogenicznych drzew SSV wyrażenie d(·,vNNN) oznacza kwadrat[3] odległości euklidesowej od wektora, który jest elementem zbioru treningowego i ma w nim indeks NNN.

Systemy heterogeniczne mają wiele zalet nie tylko wtedy, gdy są drzewami decyzji. Różnorodność wewnętrznej reprezentacji zwiększa możliwości każdego systemu klasyfikacji [53], ale też zawsze wiąże się z koniecznością szukania „złotego środka" pomiędzy bogactwem reprezentacji a atrakcyjnym czasem działania systemu.

3.3.1 Przykłady

Zalety heterogenicznych drzew decyzji pokazują wyraźnie poniższe dwa przykłady sztucznie wygenerowanych danych klasyfikacyjnych.

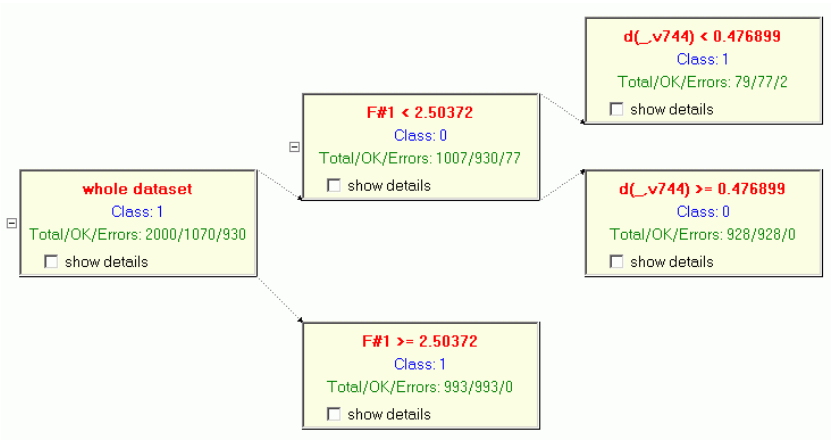
Przykład 3.1 *W przestrzeni R^2 generujemy 2000 punktów mieszczących się w kwadracie o bokach na odcinkach $[0,5]$. Wektory klasyfikujemy do dwóch klas odpowiadających prawdziwości bądź nie wyrażenia $x > 2.5 \vee (x-2.5)^2 + (y-2.5)^2 < 0.55$. Wygenerowane dane przedstawia wykres 3.4.*



Rysunek 3.4: Sztucznie wygenerowane dane dwuklasowe

Tak postawiony problem klasyfikacji jest dla klasycznych systemów drzew decyzji problemem bardzo trudnym. Drzewo SSV *w procesie szukania wiązką w 5-krotnej kroswalidacji znajduje opis danych składający się z 9 reguł, które łącznie zawierają 34 przesłanki i klasyfikują zbiór treningowy z dwoma błędami. Użycie heterogenicznej wersji* SSV *prowadzi do prostego drzewa przedstawionego na rysunku 3.5 nawet w przypadku szukania metodą „najpierw najlepszy" (zbiór treningowy jest tutaj również klasyfikowany z dwoma błędami). Reguły opisujące te dane przedstawiają się więc następująco:*

[3]Wygodniej jest porównywać kwadraty odległości zamiast odległości, bo unikamy kosztownego pierwiastkowania. Oczywiście zmiana ta nie ma wpływu na uzyskiwane wyniki.



Rysunek 3.5: Heterogeniczne drzewo SSV

$\mathcal{R}_1$: *F#1* $< 2.50372 \wedge d(\cdot,v744) < 0.476899 \rightarrow$ class 1
$\mathcal{R}_2$: *F#1* $> 2.50372 \rightarrow$ class 1
$\mathcal{R}_3$: *F#1* $< 2.50372 \wedge d(\cdot,v744) > 0.476899 \rightarrow$ class 0

gdzie wektor v744 = (2.42049, 2.54793) i jest elementem klasy 1.

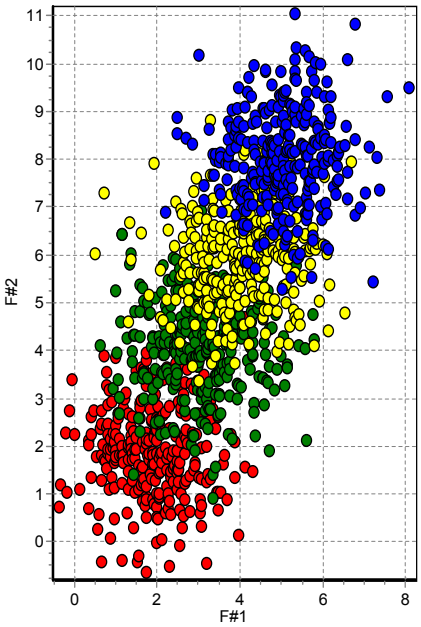
Przykład 3.2 *W przestrzeni* R^2 *generujemy cztery grupy po 300 punktów. Każda z grup ma dwuwymiarowy rozkład normalny o odchyleniach standardowych* $\sigma = 1$ *w każdym z wymiarów oraz wartości oczekiwanej odpowiednio w punkcie* $(2,2)$, $(3,4)$, $(4,6)$ *i* $(5,8)$. *Dane przedstawia rysunek 3.6(a).*

Drzewa homogeniczne produkują stosunkowo skomplikowane rozwiązania, a prostopadłe do osi podziały sa łatwo dostrzegalne na rysunku 3.6(b). Użycie drzew heterogenicznych wykorzystujące euklidesowe odległości od wektorów treningowych pozwala przy nieco lepszej dokładności klasyfikacji podzielić przestrzeń na 4 części trzema łukami, które również łatwo dostrzec na rysunku 3.6(c). Reguły klasyfikacji wyglądają w tym przypadku następująco:

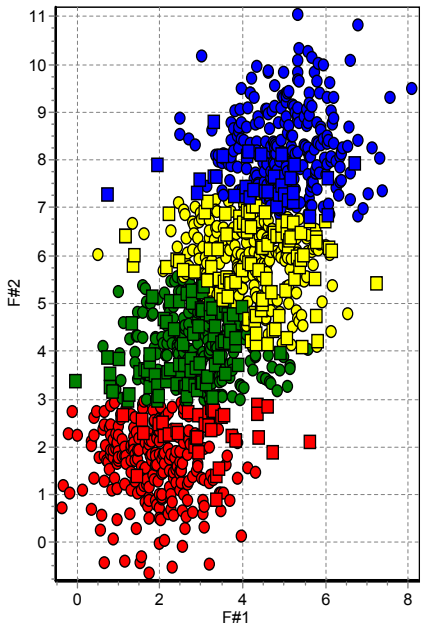
$\mathcal{R}_1$: $d(\cdot,v117) < 13.2839 \rightarrow$ class 1
$\mathcal{R}_2$: $d(\cdot,v904) > 14.8451 \wedge d(\cdot,v117) > 13.2839 \rightarrow$ class 2
$\mathcal{R}_3$: $d(\cdot,v904) < 14.8451 \wedge d(\cdot,v1179) > 13.2464 \rightarrow$ class 3
$\mathcal{R}_4$: $d(\cdot,v1179) < 13.2464 \rightarrow$ class 4

3.3.2 Rezultaty

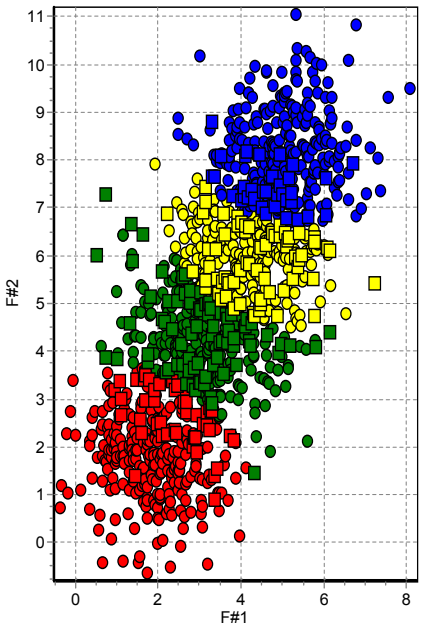
Sztucznie wygenerowane dane ilustrują zwykle dość stronniczo wyniki, jakie można otrzymać stosując dany algorytm. Aby rzetelnie ocenić metodę trzeba również



(a) Oryginalne dane



(b) Granice decyzji homogenicznego drzewa SSV



(c) Granice decyzji heterogenicznego drzewa SSV

Rysunek 3.6: Cztery klasy o rozkładach normalnych (na rys. (a) kolory odpowiadają oryginalnym klasom, na rys. (b) i (c) – klasyfikacji drzew, przy czym kwadraty oznaczają błędnie klasyfikowane wektory)

pokazać jej sprawność dla rzeczywistych problemów. Szukanie heterogenicznych drzew daje ciekawe rezultaty także dla realnych zadań klasyfikacji.

Iris. W przypadku irysów możliwie najprostsze reguły w klasycznej postaci opisujące te dane (otrzymane podstawowym algorytmem SSV) są następujące:

$\mathcal{R}_1$: petal length $< 2.45 \rightarrow$ *setosa*
$\mathcal{R}_2$: petal length $> 2.45 \wedge$ petal width $< 1.65 \rightarrow$ *virginica*
$\mathcal{R}_3$: petal length $> 2.45 \wedge$ petal width $> 1.65 \rightarrow$ *versicolor*

Ich poprawność klasyfikacji dla pełnego zbioru treningowego to 96% (6 błędów).

Heterogeniczne drzewo znalezione dla tego zbioru ma bardzo podobną postać, choć dzięki użyciu przesłanki dotyczącej odległości od pewnego punktu otrzymujemy dokładność 96.7% (5 błędów – oczywiście poprawa ta nie jest statystycznie znacząca):

$\mathcal{R}_1$: petal length $< 2.45 \rightarrow$ *setosa*
$\mathcal{R}_2$: petal length $> 2.45 \wedge$ d(_,v15) $< 16.185 \rightarrow$ *versicolor*
$\mathcal{R}_3$: petal length $> 2.45 \wedge$ d(_,v15) $> 16.185 \rightarrow$ *virginica*

Wisconsin breast cancer. Dużą niespodzianką są heterogeniczne drzewa decyzji jakie znajdują algorytmy oparte na SSV dla danych z UCI na temat raka piersi. Jednopodziałowe drzewa wykorzystujące odległość euklidesową dają bardzo dobrą jakość klasyfikacji. Proces automatycznego doboru stopnia przycinania drzewa podaje jako wynik końcowy następującą regułę:

$\mathcal{R}_1$: d(·,v303) $> 411 \rightarrow$ *benign*
$\mathcal{R}_2$: *else malignant*

Na zbiorze treningowym reguła ta popełnia 18 błędów, co daje 97.4% dokładności.

Podglądając proces uczenia przez kroswalidację możemy zaobserwować także inne bardzo podobne reguły. Na przykład:

$\mathcal{R}_1$: d(·,v279) $> 383 \rightarrow$ *benign*
$\mathcal{R}_2$: *else malignant*

albo

$\mathcal{R}_1$: d(·,v612) $> 404 \rightarrow$ *benign*
$\mathcal{R}_2$: *else malignant*.

Hypothyroid. Ponieważ w tym przypadku najlepiej spisują się systemy, które mają granice decyzji prostopadłe do osi przestrzeni, heterogeniczne drzewa nie są w stanie wnieść zbyt dużo do tego co już na temat tych danych wiadomo. Niemniej jednak można uzyskać o tyle interesujący zestaw heterogenicznych reguł, że jest on najdokładniejszy z opublikowanych dotąd rozwiązań. Reguły:

$\mathcal{R}_1$: TSH > 6.05 $\wedge$ FTI < 64.72 $\wedge$ (T3 < 11.5 $\vee$ d(·,v2917) < 1.10531) $\rightarrow$ *primary hypothyroid*

$\mathcal{R}_2$: TSH > 6.05 $\wedge$ FTI > 64.72 $\wedge$ on thyroxine = 0 $\wedge$ thyroid surgery = 0 $\wedge$ TT4 < 150.5 $\rightarrow$ *compensated hypothyroid*

$\mathcal{R}_3$: *else healthy*

klasyfikują zbiór treningowy z dokładnością 99.84% (6 błędów) oraz zbiór testowy z 99.39% (21 błędów). Jak widać przesłanka odległościowa występuje tutaj na samym końcu gałęzi, co oznacza, że jest ona brana pod uwagę tylko w małym lokalnym obszarze. Może to świadczyć o tym, że diagnozujący lekarze oceniają podobieństwa do poprzednich przypadków tylko w pewnych, szczególnych przypadkach.

Najlepsze systemy różnią się tutaj o pojedyncze błędy. W przypadku tak dużych zbiorów danych różnice te nie są statystycznie istotne, niemniej uzyskany opis regułowy wart jest uwagi.

3.4 Lasy

Zdarza się często (zwłaszcza w przypadku małych zbiorów danych, co z kolei jest bardzo powszechne w medycynie), że można stworzyć wiele różnych modeli o zbliżonej jakości klasyfikacji i podobnej złożoności. Szczególnie w przypadku systemów generujących zrozumiałe dla człowieka opisy klasyfikacji warto, by potrafiły podać nie jedno rozwiązanie, a cały ich szereg tak, by ekspert mógł otrzymać kilka alternatywnych wyjaśnień i korzystać z kilku z nich, bądź wybrać to, które jest najbardziej zgodne z jego wiedzą na dany temat.

Jedną z metod generowania zestawów różnych rozwiązań może być boosting (p. rozdział 2.13.3). Wykorzystać można w ten sposób fakt, że przy drobnych zaburzeniach rozkładów różnych klas w danych można uzyskać różne rozwiązania [22]. Jednakże czasami drobne zaburzenia rozkładu danych nie prowadzą do różnych rozwiązań i wówczas ta metoda może generować wielokrotnie to samo rozwiązanie.

Bardzo podobne wyniki do tych uzyskanych z boostingu, można otrzymać analizując poszczególne przebiegi kroswalidacji.

Obie te metody wykorzystują wariancję systemu klasyfikacji związaną z różnicami w próbkach danych treningowych. Do tego samego celu można też wykorzystać możliwości wewnętrzne systemu generującego model.

Ponieważ w przypadku drzew klasyfikacji stosuje się metody szukania, więc system w procesie uczenia zwykle analizuje wiele alternatywnych rozwiązań, by ostatecznie zdecydować się na wybór jednego z nich. Aby zapewnić sobie uzyskanie większej liczby drzew decyzji SSV, można wykorzystać algorytm szukania wiązką. Analizując jego przebieg można ocenić, które z drzew dają największe prawdopodobieństwo dobrej dokładności klasyfikacji oraz generalizacji, i w ten sposób wygenerować zestaw rozwiązań, który, jako składający się z drzew, nazywamy *lasem*.

Przedstawiona tutaj koncepcja lasu nie jest spotykana w literaturze. Co prawda, termin „las" (ang. *decision forest*) jest używany w systemie PolyAnalyst [109], jednak w całkowicie innym znaczeniu. Określa on tutaj zestawy drzew specjalizujących się w klasyfikacji poszczególnych klas – w przypadku danych *n*-klasowych lasem jest więc *n* drzew rozstrzygających problemy dwuklasowe, polegające na rozróżnianiu jednej z klas od pozostałych (taka technika stosowana jest w wielu innych systemach i jest określana zwykle nazwami „*one-per-class*" lub „*one-against-rest*").

Algorytm zastosowany do tworzenia lasów SSV jest następujący:

Algorytm 3.7 (Szukanie lasu drzew)

► ***Dane:*** *Zbiór treningowy T, liczba przebiegów kroswalidacji $n \in N$, typ kroswalidacji.*

◄ ***Wynik:*** *Lista drzew klasyfikacji.*

1. *Wykonujemy n-krotną kroswalidację dla zbioru treningowego. Dla każdego przebiegu:*
   - *Szukamy wiązką drzewa modyfikując punkt 3 algorytmu 3.2 tak, by zatrzymać proces dopiero wtedy, gdy wszystkie drzewa wiązki będą finalne.*
   - *dla każdej pary (s, p) gdzie $s \in \{1, \ldots, m\}$, m jest numerem etapu procesu szukania wiązką (liczbą wygenerowanych do danego etapu wiązek), $p \in \{1, \ldots, k_s\}$, k_s jest liczbą elementów s-tej wiązki, liczymy błąd walidacyjny p-tego drzewa s-tej wiązki $E_{WAL} = E_{TRN} + n \cdot E_{TST}$, gdzie E_{TRN} i E_{TST} są błędami klasyfikacji odpowiednio dla zbiorów treningowego i walidacyjnego.*
2. *Szukamy wiązką drzewa dla całego zbioru treningowego.*

3. *Przypisujemy p-temu drzewu s-tej wiązki średni błąd walidacyjny dla pary* (s, p)*, przy czym:*
   - *Pomijamy pary, którym przynajmniej w jednym z przebiegów kroswalidacji nie odpowiada żaden stan (opieramy się tutaj na założeniu, że stan procesu szukania, który nie wystąpił we wszystkich etapach kroswalidacji daje niskie prawdopodobieństwo dobrej generalizacji.*
   - *Przy wyliczaniu średniej pomijamy dwie skrajne wartości.*
4. *Porządkujemy drzewa, które pojawiły się we wiązkach różnych poziomów procesu szukania pod względem błędu walidacyjnego.*
5. *Uporządkowana lista drzew jest wynikiem działania algorytmu.*

W gestii użytkownika leży dobór takiej liczby drzew, które uzna za interesujące i podda dodatkowej analizie.

Warto zwrócić też uwagę na fakt, że jeśli różne zestawy reguł dają bardzo podobną poprawność klasyfikacji, to mogą różnić się wrażliwością lub znamiennością, i z tego powodu również mogą stanowić różną wartość dla eksperta i podejmowanej przez niego konkretnej diagnozy.

3.4.1 Rezultaty

Algorytm generowania lasów SSV został zastosowany do kilku medycznych problemów klasyfikacji zawartych w repozytorium UCI [126]. Zazwyczaj zestawy reguł opisujące drzewa z początku wygenerowanej listy są bardzo zbliżone do najbardziej zwięzłych i dokładnych opisów dostępnych w literaturze. Dla każdego z opisanych niżej problemów, drzewa są przedstawiane w kolejności odpowiadającej ich porządkowi w wygenerowanym lesie.

Hypothyroid. Dla zbioru chorób tarczycy na samym początku lasu przedstawiony został następujący zestaw reguł (wartości ciągłych cech zostały przemnożone przez 1000):

$\mathcal{R}_1$: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid surgery = no $\rightarrow$ *primary hypothyroid*

$\mathcal{R}_2$: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\wedge$ TT4 $<$ 150.5 $\rightarrow$ *compensated hypothyroid*

$\mathcal{R}_3$: *else healthy*

Reguły te są poprawne w 99.79% (8 błędów) dla zbioru treningowego oraz w 99.33% (23 błędy) dla zbioru testowego.

Drugą pozycję w lesie zajmuje następujące drzewo o dokładności 99.76% (9 błędów) na zbiorze treningowym i 99.36% (22 błędy) na zbiorze testowym:

$\mathcal{R}_1$: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\rightarrow$ *primary hypothyroid*
$\mathcal{R}_2$: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\wedge$ TT4 $<$ 150.5 $\rightarrow$ *compensated hypothyroid*
$\mathcal{R}_3$: *else healthy*

Te same merytorycznie reguły pojawiają się po kilka razy w lesie, ponieważ powstają z równoważnych drzew o nieco różnych postaciach. Na piątej pozycji listy znajdujemy nieco inne drzewo (troszkę mniej złożone) o dokładnościach odpowiednio 99.52% oraz 98.89%:

$\mathcal{R}_1$: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid surgery = no $\rightarrow$ *primary hypothyroid*
$\mathcal{R}_2$: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\rightarrow$ *compensated hypothyroid*
$\mathcal{R}_3$: *else healthy*

Na dalszych pozycjach listy spotykamy jeszcze prostsze drzewa. Na przykład na pozycji 13-tej drzewo o dokładności treningowej 99.50% i testowej 98.92%:

$\mathcal{R}_1$: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\rightarrow$ *primary hypothyroid*
$\mathcal{R}_2$: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\rightarrow$ *compensated hypothyroid*
$\mathcal{R}_3$: *else healthy*

Wisconsin breast cancer. Niektóre z zestawów reguł reprezentujących drzewa z wygenerowanego lasu przedstawiają się następująco (obok poprawności, w nawiasach, widnieją liczby błędów oraz wektorów niesklasyfikowanych):

$\mathcal{R}_1$: $F_3 < 2.5 \rightarrow$ *benign*
$\mathcal{R}_2$: $F_6 < 2.5 \wedge F_5 < 3.5 \rightarrow$ *benign*
$\mathcal{R}_3$: *else malignant*

95.6% poprawności (25 + 6)
95.0% wrażliwości
95.9% znamienności

$\mathcal{R}_1$: $F_3 < 2.5 \rightarrow$ *benign*
$\mathcal{R}_2$: $F_5 < 2.5 \wedge F_8 < 2.5 \rightarrow$ *benign*
$\mathcal{R}_3$: *else malignant*

95.0% poprawności (35 + 0)
93.8% wrażliwości
95.6% znamienności

$\mathcal{R}_1$: $F_3 < 2.5 \rightarrow$ *benign*
$\mathcal{R}_2$: $F_5 < 2.5 \wedge F_2 < 2.5 \rightarrow$ *benign*
$\mathcal{R}_3$: *else malignant*

95.1% poprawności (34 + 0)
95.9% wrażliwości
94.8% znamienności

$\mathcal{R}_1$: $F_3 < 2.5 \rightarrow$ *benign* $\mathcal{R}_2$: $F_5 < 2.5 \wedge F_6 < 2.5 \rightarrow$ *benign* $\mathcal{R}_3$: *else malignant*	95.1% poprawności (34 + 0) 95.0% wrażliwości 95.2% znamienności
$\mathcal{R}_1$: $F_2 < 2.5 \rightarrow$ *benign* $\mathcal{R}_2$: $F_2 < 4.5 \wedge F_6 < 3.5 \rightarrow$ *benign* $\mathcal{R}_3$: *else malignant*	95.0% poprawności (33 + 2) 90.5% wrażliwości 97.4% znamienności
$\mathcal{R}_1$: $F_3 < 2.5 \rightarrow$ *benign* $\mathcal{R}_2$: $F_6 < 2.5 \wedge F_3 < 4.5 \rightarrow$ *benign* $\mathcal{R}_3$: *else malignant*	95.1% poprawności (34 + 0) 94.2% wrażliwości 95.6% znamienności

Ljubljana breast cancer. Prawdopodobnie z powodu bardzo małolicznej próbki danych otrzymujemy tutaj na pierwszej pozycji listy drzewo, które ma dokładność 75.5% (70 błędów), wrażliwość 30.5% oraz znamienność 94.5%:

$\mathcal{R}_1$: breast = left $\wedge$ inv nodes $> 2.5 \rightarrow$ *recurrence-events*
$\mathcal{R}_2$: *else no-recurrence-events*

Trudno jednak przypisywać temu zestawowi reguł poważny bagaż niesionej informacji (ze względu na przesłankę „breast = left” można wnioskować, że stosunkowo duża reprezentatywność tych reguł wiąże się z ubogością zbioru treningowego).

Inne drzewo daje reguły klasyfikacji, których zrozumienie nie wymaga nawet znajomości wiedzy medycznej:

$\mathcal{R}_1$: deg-malig $> 2.5 \wedge$ inv-nodes $> 2.5 \rightarrow$ *recurrence-events*
$\mathcal{R}_2$: *else no-recurrence-events*

Jest to ten sam zestaw, który został uzyskany klasycznym algorytmem (strona 6). Reguły te klasyfikują błędnie 68 przypadków, mają wrażliwość 31.8% oraz znamienność 95.0%.

Las proponuje nam także inne zestawy reguł (nieco dokładniejsze i bardziej złożone), jednak nie należy spodziewać się po nich dobrego uogólnienia problemu. Jednym z takich zestawów jest następujący (66 błędów, 47.1% wrażliwość, 89.6% znamienność):

$\mathcal{R}_1$: deg-malig $> 2.5 \wedge$ inv-nodes $> 2.5 \rightarrow$ *recurrence-events*
$\mathcal{R}_2$: deg-malig $> 2.5 \wedge$ inv-nodes $< 2.5 \wedge$ (tumor-size = 25-34 $\vee$ tumor-size = 50-54) $\rightarrow$ *recurrence-events*
$\mathcal{R}_3$: *else no-recurrence-events*

Reguły	Dokładność / Wrażliwość / Znamienność
$\mathcal{R}_1$: d(·,v73) < 12.0798 ∨ MBAA > 1174.5 → *class 1* $\mathcal{R}_2$: *else class 2*	92.5% / 98.8% / 66.7%
$\mathcal{R}_1$: d(·,v22) < 16.4115 ∨ MBAP > 12 → *class 1* $\mathcal{R}_2$: *else class 2*	92.5% / 98.8% / 66.7%
$\mathcal{R}_1$: d(·,v22) < 16.4115 ∨ MBAA > 1174.5 → *class 1* $\mathcal{R}_2$: *else class 2*	92.5% / 98.8% / 66.7%
$\mathcal{R}_1$: d(·,v8) < 13.385 ∨ MBAA > 1174.5 → *class 1* $\mathcal{R}_2$: *else class 2*	92.5% / 98.8% / 66.7%
$\mathcal{R}_1$: d(·,v8) < 13.385 ∧ HNEA $\notin (9543, 9997)$ → *class 1* $\mathcal{R}_2$: d(·,v8) > 13.385 ∧ MBAA > 1174.5 ∧ MNEP > 51 → *class 1* $\mathcal{R}_3$: *else class 2*	96.2% / 98.8% / 85.7%

Tabela 3.14: Heterogeniczny las dla danych Appendicitis

3.5 Lasy heterogeniczne

Użycie algorytmu tworzenia lasów jednocześnie z techniką heterogenicznych przesłanek prowadzi do heterogenicznych lasów. Należy jednak mieć na względzie, że w ten sposób otrzymamy algorytm o bardzo dużej (w porównaniu z oryginalną metodą szukania drzew) złożoności.

3.5.1 Rezultaty

Podobnie jak w przypadku zwykłych lasów tak i tutaj zestawy reguł prezentowane są w kolejności, która odpowiada ich porządkowi w lesie.

Appendicitis. Heterogeniczne lasy odkryły kilka nowych opisów dla tego zbioru danych, które są proste, dokładne i bardzo wrażliwe. Kilka zestawów reguł przedstawia tabela 3.14. Przesłanki dotyczące odległości uwzględniają standaryzację danych.

Reguły	Dokładność / Wrażliwość / Znamienność
$\mathcal{R}_1$: d(·,v303) < 62.7239 → *malignant* $\mathcal{R}_2$: *else benign*	97.3% / 97.9% / 96.9%
$\mathcal{R}_1$: d(·,v681) < 51.9701 → *malignant* $\mathcal{R}_2$: *else benign*	97.4% / 98.8% / 96.8%
$\mathcal{R}_1$: d(·,v613) < 65.3062 → *malignant* $\mathcal{R}_2$: *else benign*	97.1% / 97.9% / 96.7%
$\mathcal{R}_1$: d(·,v160) < 36.4661 → *malignant* $\mathcal{R}_2$: *else benign*	97.1% / 98.8% / 96.3%
$\mathcal{R}_1$: d(·,v150) < 39.5778 → *malignant* $\mathcal{R}_2$: *else benign*	97.1% / 98.8% / 96.3%
$\mathcal{R}_1$: d(·,v613) < 65.3062 → *malignant* $\mathcal{R}_2$: *else benign*	97.1% / 97.9% / 96.7%

Tabela 3.15: Heterogeniczny las dla danych Wisconsin

Wisconsin breast cancer. Ponieważ dla tych danych jednoprzesłankowe reguły wykorzystujące odległości są zdecydowanie prostsze niż reguły klasycznej postaci, więc przsłanki odległościowe całkowicie wyparły te klasyczne i otrzymujemy las przedstawiony w tabeli 3.15.

3.6 Dyskretyzacja

Niektóre metody klasyfikacji (jak np. te oparte na teorii zbiorów przybliżonych [142, 143]) są zaprojektowane tak, że działają tylko dla danych dyskretnych. W takich sytuacjach pełne systemy klasyfikacji zawierają w sobie moduły, odpowiadające za przetworzenie danych o charakterze ciągłym w dane symboliczne (taką konwersję nazywamy *dyskretyzacją*). Również do tego celu można zastosować kryterium separowalności SSV.

Celem dyskretyzacji jest dokonanie podziału cechy ciągłej na rozłączne i pokrywające jej zakres odcinki tak, by przynależność do poszczególnych odcinków zawierała w sobie jak najwięcej informacji o klasach obiektów. Punkty, które dzielą zbiór liczb rzeczywistych na odcinki powinny być więc tak dobrane, żeby jak najlepiej oddzielały od siebie wektory z różnych klas. Zatem zadanie to jest zbieżne z zadaniem realizowanym przez kryterium separowalności SSV.

Poza zastosowaniami do przygotowywania danych dla systemów klasyfikacji działających w przestrzeniach cech symbolicznych, dyskretyzacja może być także

skutecznie wykorzystana jako część metody selekcji cech (p. rozdział 3.8).

Algorytm 3.8 przedstawia sposób wykorzystania kryterium SSV oraz drzew decyzji dla celu dyskretyzacji.

Algorytm 3.8 (Dyskretyzacja cechy ciągłej)

► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$, cecha ciągła F przestrzeni X, maksymalna liczba podziałów $n \in N$, parametry metody szukania drzewa.*

◄ ***Wynik:*** *Zbiór punktów podziału.*

1. *Budujemy drzewo decyzji D dla zbioru treningowego $T_F = \{(x_F, c) \in F \times C : (x, c) \in T$ gdzie x_F jest wartością cechy F dla wektora x. Korzystamy z algorytmu 3.1, 3.2 bądź 3.3 zgodnie z zadanymi parametrami.*

2. *Przycinamy drzewo D podobnie jak w algorytmie 3.5, ale z tą różnicą, że zamiast liczby liści kontrolujemy liczbę węzłów drzewa nie będących liśćmi.*

3. *Wynikiem jest zbiór złożony ze wszystkich punktów podziału występujących w przyciętym drzewie D.*

Powyższa metoda dyskretyzuje pojedynczą cechę przestrzeni klasyfikacji. Można ją zastosować niezależnie dla każdej ciągłej cechy przestrzeni i w ten sposób stworzyć odpowiednią przestrzeń cech dyskretnych. Niezależność podziałów różnych cech może sprawiać, że powstała w ten sposób przestrzeń dyskretna nie pozwala na klasyfikację danych z podobną poprawnością jak oryginalna przestrzeń. Taka sytuacja może mieć miejsce wtedy, gdy separowalność klas można osiągnąć jedynie dobierając jednocześnie odpowiednie podziały dla różnych cech, bo każdy z wymiarów z osobna nie daje możliwości rozseparowania klas (np. dla danych ułożonych w R^2 w sposób przypominający szachownicę). Można stworzyć algorytm dyskretyzacji, który uwzględni także punkty podziału korzystne z punktu widzenia klasyfikacji w podprzestrzeniach dwuwymiarowych (czy nawet większej wymiarowości). Rozwiązanie takie może jednak doprowadzić do sytuacji, w której dyskretyzacja niejako sugeruje rozwiązanie ostatecznemu modelowi. Nie byłoby na przykład sensowne, żeby dyskretyzacja odzwierciedlała podziały drzewa zbudowanego dla pełnej przestrzeni cech, bo wówczas liczba podziałów byłaby zwykle na tyle mała, że w powstałej przestrzeni możliwe byłoby pełne przeszukanie zbioru możliwych rozwiązań, a to najczęściej prowadziłoby do odtworzenia klasyfikatora reprezentowanego przez drzewo.

Uwaga : Kiedy jednym z etapów działania systemu klasyfikacji jest dyskretyzacja, należy zwracać uwagę na to, by była ona wykonywana w sposób nie zniekształcający wyników badań. Nie można na przykład wykonywać dyskretyzacji na etapie przygotowywania danych do testu kroswalidacyjnego, a należy wykonywać ją niezależnie w każdym z przebiegów testu. Jak poważnie mogą w ten sposób zostać wypaczone wyniki może świadczyć przypadek dyskretyzacji zbioru *Appendicitis*, w którym po dyskretyzacji kroswalidacja drzewa SSV daje wynik o ok. 5% lepszy niż na oryginalnych danych (a dla tego zbioru danych jest to różnica kolosalna). Jest to jednak ewidentne wykorzystywanie zbioru testowego na etapie trenowania modelu, co oczywiście jest poważnym nadużyciem.

3.6.1 Rezultaty

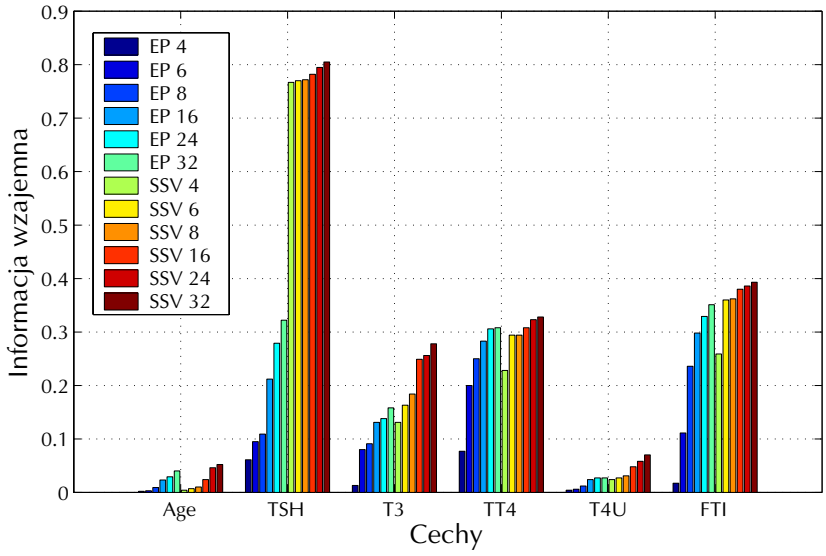
Jednym ze sposobów oceny jakości dyskretyzacji jest ocena ilości informacji o klasach, jaką mają w sobie powstałe przedziały. Podstawowym kryterium przydatności metody dyskretyzacji jest większy bagaż informacji niż ma w sobie podział zakresu danych na równe części.

Hypothyroid. Dyskretyzacja sześciu cech ciągłych, które obok piętnastu binarnych opisują dane *Hypothyroid*, wykonana poprzez drzewo SSV pokazuje, że podziały dokonane przez drzewo są bardzo trafne. Porównanie wartości informacji wzajemnej [61] pomiędzy zdyskretyzowanymi cechami a klasą dla podziałów na przedziały równej długości (ang. *equiwidth partitioning* – EP) oraz uzyskanych metodą opartą na SSV przedstawia rysunek 3.7. Z rysunku widać, że już przy podziale na 4 części dyskretyzacja SSV może dać zdecydowanie lepsze wyniki niż podział nawet na 32 równe części (dla cechy TSH podział na 4 części przy użyciu SSV daje wartość informacji wzajemnej równą 0.767 podczas gdy podział na 32 równe części daje tylko 0.322).

3.7 Uciąglanie

W przypadku wielu systemów klasyfikacji (zwłaszcza tych opartych na odległościach między wektorami) dane wejściowe muszą mieć postać numeryczną. Aby takie systemy stosować do danych symbolicznych potrzeba więc metody o działaniu odwrotnym do dyskretyzacji, czyli konwersji cech dyskretnych w numeryczne. Taką operację można nazwać *uciągleniem* cechy dyskretnej. Polega ona na przypisaniu liczb rzeczywistych poszczególnym symbolom ze zbioru wartości cechy.

Najprostszym sposobem (i chyba najczęściej stosowanym w różnych metodach) jest przypisanie symbolom kolejnych liczb naturalnych w sposób arbitralny.



Rysunek 3.7: Porównanie ilości informacji dla dyskretyzacji dzielącej na równe przedziały (EP) oraz opartej na SSV (liczby obok nazw metod to liczby przedziałów powstałych w wyniku dyskretyzacji)

Jednak w takiej sytuacji uporządkowanie symboli jest zupełnie przypadkowe, co najczęściej skutkuje tym, że z tak przetworzonych danych, żaden system CI nie jest w stanie wyciągnąć maksymalnej ilości informacji.

Niektóre systemy klasyfikacji stosują technikę zastępowania cech symbolicznych kilkoma cechami binarnymi. Jeśli cecha może przyjmować n różnych symboli, to zastępuje się ją zestawem n cech binarnych (ang. *n binary features* – NBF) takich, że dla $i = 1,\ldots,n$, i-ta z nowych cech wskazuje czy dla danego wektora wartość wyjściowej cechy jest i-tym symbolem, czy nie. Taka konwersja jest na ogół zdecydowanie skuteczniejsza niż arbitralne przypisywanie liczb naturalnych, ale może poważnie zwiększać wymiar przestrzeni klasyfikacji.

W systemach minimalnoodległościowych, do liczenia odległości pomiędzy danymi dyskretnymi, stosuje się pewne specjalne miary, takie jak Value Difference Metric (VDM), Heterogeneous Euclidean-Overlap Metric [183] czy Minimum Risk Metric [10].

VDM jest miarą, która z powodzeniem może być wykorzystana do zastąpienia cech dyskretnych ciągłymi [54]. W przestrzeni cech dyskretnych $X = X_1 \times \cdots, X_n$ dla zbioru klas $C = \{c_1,\ldots,c_k\}$ definiuje się dla $x,y \in X \times C$

$$D^q_{\mathsf{VDM}}(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{k} |P(c_j|X_i = x_i) - P(c_j|X_i = y_i)|^q \tag{3.14}$$

gdzie $P(c_j|X_i = z_i)$ jest skrótem zapisu $P(C(u) = c_j|u_i = z_i \wedge u \in X \times C)$.

W [54] autorzy przedstawili pomysł zastąpienia dla każdego wektora x wartości x_i cechy dyskretnej X_i przez k wartości ciągłych $P(c_1|X_i = x_i), \ldots, P(c_k|X_i = x_i)$. W problemach dwuklasowych wystarczy dodać wartość dla jednej z klas i w ten sposób nie zwiększa się wymiarowości danych. W przypadkach wieloklasowych niestety wymiar przestrzeni klasyfikacji może znacznie wzrosnąć.

Naturalną alternatywą dla metody VDM jest odwrócenie warunków w wykorzystywanym tam prawdopodobieństwie warunkowym [76]. Zastępując $P(c_j|X_i = x_i)$ przez $P(X_i = x_i|c_j)$ otrzymujemy odwzorowanie podobne w idei do VDM (dlatego ta metoda otrzymała nazwę MDV), aczkolwiek pod pewnymi względami znacznie się różniące.

Główną różnicą jest fakt, że VDM porządkuje symbole według ich specyficzności dla danej klasy, natomiast MDV według częstości występowania w zbiorze wektorów z danej klasy.

Natura VDM powoduje, że w problemach dwuklasowych wystarczy użyć jednego z prawdopodobieństw, natomiast dla MDV prawdopodobieństwa wyliczone dla klas nie są od siebie tak zależne, choć w szczególnych przypadkach można podobną zależność zaobserwować (np. wtedy, gdy występowanie każdego z symboli w całym zbiorze danych jest jednakowo częste, bo wówczas kolejności wynikające z częstości występowania symboli dla dwóch klas są wzajemnie odwrotne). Dla zredukowania wymiarowości przestrzeni można, mimo to, ograniczyć liczbę cech uwzględniając prawdopodobieństwa tylko dla jednej z klas.

W problemach wieloklasowych (więcej niż dwu) zdefiniowanych w przestrzeniach cech dyskretnych, metody VDM oraz MDV stworzą w wyniku konwersji dane w przestrzeni o wymiarowości równej iloczynowi wymiaru przestrzeni źródłowej i liczby klas, natomiast metoda NBF w przestrzeni o wymiarowości równej sumie liczby możliwych wartości wszystkich cech. We wszystkich tych przypadkach wymiarowość może bardzo znacznie wzrosnąć, co zwykle ma niekorzystny wpływ na czas trwania procesów adaptacji modeli CI, oraz nie sprzyja skuteczności większości metod.

Na bazie kryterium SSV można stworzyć metodę zamiany wartości symbolicznych na rzeczywiste pozbawioną tej wady. Przedstawia ją algorytm 3.9.

Algorytm 3.9 (Uciąglenie cechy dyskretnej)

- ► ***Dane:*** *Przestrzeń klasyfikacji X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$, cecha dyskretna F przestrzeni X, parametry metody szukania drzewa.*
- ◄ ***Wynik:*** *Odwzorowanie $F \to R$.*

1. *Budujemy drzewo decyzji D dla zbioru treningowego $T_F = \{(x_F, c) \in F \times C : (x, c) \in T\}$ gdzie x_F jest wartością cechy F dla wektora x. Korzystamy z algorytmu 3.1, 3.2 bądź 3.3 zgodnie z zadanymi parametrami.*

2. *Dla każdego węzła W drzewa D, które nie jest korzeniem ani podwęzłem korzenia wyznaczamy wartość SSV_W jako wartość kryterium* SSV *dla podziału pomiędzy węzłem W a bratem jego rodzica (rozpatrujemy podział zbioru wektorów treningowych wpadających do tych węzłów na dwa podbiory zdeterminowane właśnie tymi węzłami).*

3. *Dla każdego węzła W drzewa D, które nie jest korzeniem ani liściem (zaczynając od bezpośrednich podwęzłów korzenia, poprzez ich podwęzły aż do nadwęzłów liści) porządkujemy jego dzieci W_i:*

   - *Według malejących od lewej do prawej wartości SSV_{W_i} jeśli W jest lewym dzieckiem swojego rodzica.*
   - *Według rosnących od lewej do prawej wartości SSV_{W_i} jeśli W jest prawym dzieckiem swojego rodzica.*

4. *Tworzymy listę $L_1,\ldots,L_n$ liści drzewa ustawiając je w kolejności odwiedzania metodą szukania w głąb (przy kolejności wchodzenia w podwęzły od lewej do prawej).*

5. *Liczymy wartość kryterium $\mathsf{SSV}_{i,i+1}$ dla par liści sąsiadujących ze sobą w stworzonej liście ($i = 1,\ldots,n-1$).*

6. *Dla każdego $i = 1,\ldots,n$ przypisujemy liściowi L_i wartość rzeczywistą*

$$\frac{\sum_{j=1}^{i-1} \mathsf{SSV}_{j,j+1}}{\sum_{j=1}^{n-1} \mathsf{SSV}_{j,j+1}}. \tag{3.15}$$

7. *Wynikiem algorytmu jest odwzorowanie, które każdej wartości cechy F przyporządkowuje liczbę rzeczywistą wyznaczoną w poprzednim punkcie dla liścia, do którego wpadają wektory z tą wartością.*

Uwagi :

1. Algorytm wykorzystuje własność, że drzewo SSV skonstruowane dla cechy dyskretnej umieści zazwyczaj każdą wartość dyskretną w osobnym liściu. Inaczej może być tylko wtedy, gdy wszystkie wektory mające jedną z kilku wartości należą do tej samej klasy – wówczas z punktu widzenia klasyfikacji te wartości cechy mogą zostać ze sobą utożsamione, co zostanie dokonane przez powyższy algorytm wyznaczonym odwzorowaniem.

2. Algorytm jest bardzo efektywny, bowiem maksymalna liczba węzłów drzewa zbudowanego dla pojedynczej cechy jest o 1 mniejsza niż liczba wartości danej cechy.

3. Podobnie jak w przypadku dyskretyzacji, tak i przy uciąglaniu należy pamiętać o tym, że zastosowane jako metoda przygotowania danych dla późniejszych testów takich jak kroswalidacja prowadzi do nadmiernie optymistycznych rezultatów.

3.7.1 Rezultaty

Testowanie metod uciąglania danych może być wykonywane tylko na takich bazach, gdzie jest stosunkowo dużo cech symbolicznych. Nie ma najmniejszego sensu uciąglanie cech binarnych, wobec czego dane które są opisane wyłącznie cechami ciągłymi bądź binarnymi nie nadają się do testów tych metod. Wyciąganie wniosków na podstawie wyników uzyskanych dla danych o małej liczbie cech dyskretnych również nie jest uzasadnione.

Z tego powodu poniższe porównanie metod zostało przeprowadzone dla jednego sztucznie sporządzonego testu, oraz dla trzech zbiorów danych dostępnych w UCI, zdefiniowanych w przestrzeniach cech dyskretnych: *Promoters*, *Soybean* oraz *DNA*.

Wyniki przedstawione dla danych *Promoters* w tabeli 3.16 oraz dla danych *Soybean* w tabeli 3.17 uzyskane zostały przez powtórzenie 10 razy 10-krotnej kroswalidacji. Dane *DNA* są podzielone na części treningową i testową, więc tabela 3.18 przedstawia uśrednione wyniki 10-krotnego testowania modeli na części testowej danych po uprzednim uczeniu na części treningowej.

Każda z tabel składa się z trzech części, przedstawiających wyniki odpowiednio dla trzech modeli adaptacyjnych: sieci neuronowej FSM [50, 1], modelu SVM (w wersji Platt'a o nazwie Sequential Minimal Optimization (SMO) [145] z dodatkami, które zaproponował Keerthi [107]) oraz modelu *k* najbliższych sąsiadów (kNN) z automatycznym doborem parametru *k* na podstawie uczenia przez kroswalidację.

Dla każdego z modeli poszczególne wiersze przedstawiają wyniki uzyskane metodą przygotowania danych określoną w pierwszej kolumnie. Skrót *Arb.* oznacza arbitralne kodowanie symboli (brak specjalnej metody uciąglania). Kody *k VDM* oraz *k MDV* oznaczają odpowiednie metody uciąglania stosowane wewnątrz komitetu *k* modeli (gdzie *k* jest równe liczbie klas), z których każdy uczony jest rozróżniania jednej z klas od pozostałych.

Druga kolumna przedstawia liczbę cech przestrzeni klasyfikacji stworzonej w wyniku uciąglania.

Kolumny $P_1, P_2, \ldots$ pokazują, dla każdej z metod, prawdopodobieństwa, że jej średnia poprawność testowa jest wyższa niż uzyskana metodą odpowiednio $1, 2,$ itd. Prawdopodobieństwa te zostały oszacowane przy użyciu testu t Studenta dla różnic zmiennych skorelowanych jako, że różnice poprawności były liczone dla modeli uczonych i testowanych na tych samych parach zbiorów. Pogrubioną

czcionką wyróżnione są prawdopodobieństwa wyższe niż powszechnie stosowany próg statystycznej istotności (0.95).

Test odtworzenia poprawnego porządku

Podstawowym testem jaki powinna przejść każda metoda uciąglania danych jest zdolność odtworzenia poprawnego porządku symboli w przypadkach oczywistych dla człowieka. Generujemy więc losowo wielowymiarowe dane w przestrzeni cech symbolicznych, z których każda jest zbiorem trzech wartości (a, b oraz c). W każdym wymiarze losowanie odbywa się niezależnie zgodnie z następującymi rozkładami: dla jednej z klas $P(a) = 0.6, P(b) = 0.4, P(c) = 0$, a dla drugiej $P(a) = 0, P(b) = 0.4, P(c) = 0.6$. Tworzymy tyle wymiarów ile jest permutacji trzech symboli a, b i c takich, że porządek nie odpowiada naturalnemu układowi rozdzielającymi klasy tj. takiemu, że b znajduje się pośrodku. Takich permutacji jest $3! - 2 = 4$, uzyskujemy więc 4-wymiarową przestrzeń. Arbitralne kodowanie każdej cechy przypisuje symbolom liczby $0, 1, 2$ odpowiednio do pozycji w danej permutacji.

Wykonane testy potwierdzają, że metody VDM, MDV oraz SSV tworzą poprawną kolejność symboli i w teście 10-krotnej kroswalidacji, która w każdym kroku wykonuje najpierw konwersję danych a potem klasyfikację metodą k najbliższych sąsiadów z parametrem $k = 1$ dają 100% poprawność. Ta sama metoda klasyfikacji z uciąglaniem NBF uzyskuje 98.3% poprawności, natomiast bez uciąglania danych (z arbitralnym kodowaniem) – 96.7%.

Promoters

Zbiór danych *Promoters* zawiera 106 wektorów opisanych w przestrzeni 57 symbolicznych atrybutów, z których każdy przyjmuje wartości A, C, G i T. Dane reprezentują dwie klasy (dana sekwencja DNA zawiera lub nie zawiera fragmenty inicjujące gen), po 53 wektory każda. Wyniki testu porównawczego przedstawia tabela 3.16.

Soybean

Zbiór danych *Soybean* składa się z 307 wektorów reprezentujących 19 klas. W związku z tym, że pewne klasy mają zaledwie kilku reprezentantów testy zostały przeprowadzone na zbiorze składającym się z 290 wektorów reprezentujących 15 klas, uzyskanym przez usunięcie wektorów z czterech najmniej licznych klas (mających maksymalnie 6 wektorów). Przestrzeń klasyfikacji jest iloczynem kartezjańskim 35 symbolicznych cech, spośród których 17 to cechy binarne, jednak pozostałe 18 cech pozwala na zaobserwowanie istotnej poprawy w wynikach

FSM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5
1: Arb.	57	0.673	0.034	—	0.000	0.000	0.000	0.000
2: SSV	57	0.878	0.029	**1.000**	—	0.175	0.308	0.540
3: NBF	228	0.912	0.013	**1.000**	0.825	—	0.734	**0.968**
4: VDM	57	0.893	0.024	**1.000**	0.692	0.266	—	0.706
5: MDV	57	0.874	0.017	**1.000**	0.460	0.032	0.294	—
SVM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5
1: Arb.	57	0.478	0.014	—	0.000	0.000	0.000	0.000
2: SSV	57	0.903	0.015	**1.000**	—	**1.000**	0.091	0.045
3: NBF	228	0.695	0.040	**1.000**	0.000	—	0.000	0.000
4: VDM	57	0.930	0.016	**1.000**	0.909	**1.000**	—	0.376
5: MDV	57	0.936	0.006	**1.000**	**0.955**	**1.000**	0.624	—
kNN	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5
1: Arb.	57	0.725	0.026	—	0.001	0.069	0.000	0.000
2: SSV	57	0.860	0.020	**0.999**	—	**0.992**	0.008	0.014
3: NBF	228	0.771	0.022	0.931	0.008	—	0.000	0.000
4: VDM	57	0.929	0.019	**1.000**	**0.992**	**1.000**	—	0.587
5: MDV	57	0.924	0.011	**1.000**	**0.986**	**1.000**	0.413	—

Tabela 3.16: Wyniki testu uciąglania danych *Promoters*

uzyskanej dzięki zastosowaniu metod uciąglania. Wyniki testu porównawczego przedstawia tabela 3.17.

DNA

Zbiór *DNA* (zaczerpnięty z Genbank 64.1 – ftp://genbank.bio.net) jest podzielony na części treningową (2000 wektorów) i testową (1186 wektorów). Przestrzeń klasyfikacji to przestrzeń 60 cech symbolicznych, z których każda przyjmuje wartości A, C, G i T. Dodatkowo w niektórych cechach pojawiają się nieliczne inne symbole, które odpowiadają możliwości wystąpienia jednego z kilku symboli podstawowych. Dane reprezentują trzy klasy (sekwencje zawierają przejścia exon-intron, przejścia intron-exon lub nie zawierają żadnych z nich). Wyniki testu porównawczego przedstawia tabela 3.18.

Wnioski

Wyniki pokazują, że w przypadku wielowymiarowych danych opisanych w przestrzeniach cech dyskretnych metody uciąglania przynoszą bardzo istotną poprawę wyników klasyfikacji niezależnie od rodzaju użytego klasyfikatora. Choć w przypadku danych *Soybean* i modelu *FSM* zmiana jest nieznaczna, to jednak we wszystkich pozostałych z badanych przypadków obserwujemy znaczącą poprawę.

Sposób uciąglania oparty na kryterium SSV daje średnio wyniki nieco gorsze niż VDM czy MDV, jednak ma on tę zaletę, że nie zwiększa wymiarowości przestrzeni klasyfikacji niezależnie od liczby klas i liczby wartości poszczególnych

FSM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5
1: Arb.	35	0.868	0.010	—	0.539	0.012	0.434	0.261
2: SSV	35	0.867	0.007	0.461	—	0.000	0.407	0.198
3: NBF	97	0.894	0.006	**0.988**	**1.000**	—	**0.987**	0.926
4: VDM	525	0.870	0.010	0.566	0.593	0.013	—	0.320
5: MDV	525	0.877	0.013	0.739	0.802	0.074	0.680	—

SVM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: Arb.	35	0.664	0.007	—	0.000	0.000	0.000	0.845	**1.000**	0.000
2: SSV	35	0.762	0.007	**1.000**	—	0.001	**0.999**	**1.000**	**1.000**	0.001
3: NBF	97	0.787	0.007	**1.000**	**0.999**	—	**1.000**	**1.000**	**1.000**	0.134
4: VDM	525	0.729	0.007	**1.000**	0.001	0.000	—	**1.000**	**1.000**	0.000
5: MDV	525	0.656	0.005	0.155	0.000	0.000	0.000	—	**1.000**	0.000
6: k VDM	15×35	0.487	0.007	0.000	0.000	0.000	0.000	0.000	—	0.000
7: k MDV	15×35	0.796	0.005	**1.000**	**0.999**	0.866	**1.000**	**1.000**	**1.000**	—

kNN	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: Arb.	35	0.831	0.006	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	35	0.894	0.005	**1.000**	—	0.039	0.003	0.011	**0.953**	0.517
3: NBF	97	0.909	0.005	**1.000**	**0.961**	—	0.059	0.415	**0.992**	0.942
4: VDM	525	0.923	0.007	**1.000**	**0.997**	0.941	—	0.938	**0.999**	**0.999**
5: MDV	525	0.910	0.004	**1.000**	**0.989**	0.585	0.062	—	**0.991**	**0.958**
6: k VDM	15×35	0.878	0.008	**1.000**	0.047	0.008	0.001	0.009	—	0.086
7: k MDV	15×35	0.894	0.008	**1.000**	0.483	0.058	0.001	0.042	0.914	—

Tabela 3.17: Wyniki testu uciąglania danych *Soybean*

FSM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: Arb.	60	0.906	0.007	—	0.001	0.000	0.000	0.000	0.000	0.000
2: SSV	60	0.936	0.004	**0.999**	—	0.058	0.007	0.060	0.005	0.099
3: NBF	240	0.948	0.005	**1.000**	0.942	—	0.620	0.715	0.238	0.500
4: VDM	180	0.946	0.002	**1.000**	**0.993**	0.380	—	0.699	0.052	0.405
5: MDV	180	0.944	0.003	**1.000**	0.940	0.285	0.301	—	0.032	0.282
6: k VDM	3×60	0.953	0.003	**1.000**	**0.995**	0.762	0.948	**0.968**	—	0.721
7: k MDV	3×60	0.948	0.007	**1.000**	0.901	0.500	0.595	0.718	0.279	—

SVM	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: Arb.	60	0.611	0.000	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	60	0.927	0.000	**1.000**	—	**1.000**	0.000	0.000	0.000	**1.000**
3: NBF	240	0.633	0.000	**1.000**	0.000	—	0.000	0.000	0.000	0.000
4: VDM	180	0.948	0.000	**1.000**	**1.000**	**1.000**	—	**1.000**	**1.000**	**1.000**
5: MDV	180	0.947	0.000	**1.000**	**1.000**	**1.000**	0.000	—	**1.000**	**1.000**
6: k VDM	3×60	0.935	0.000	**1.000**	**1.000**	**1.000**	0.000	0.000	—	**1.000**
7: k MDV	3×60	0.895	0.000	**1.000**	0.000	**1.000**	0.000	0.000	0.000	—

kNN	Cechy	Popr.	Odch.st.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: Arb.	60	0.676	0.001	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	60	0.876	0.003	**1.000**	—	**1.000**	0.000	0.000	0.000	0.000
3: NBF	240	0.827	0.004	**1.000**	0.000	—	0.000	0.000	0.000	0.000
4: VDM	180	0.949	0.001	**1.000**	**1.000**	**1.000**	—	0.883	0.000	0.889
5: MDV	180	0.947	0.003	**1.000**	**1.000**	**1.000**	0.117	—	0.001	0.822
6: k VDM	3×60	0.958	0.001	**1.000**	**1.000**	**1.000**	**1.000**	**0.999**	—	**0.991**
7: k MDV	3×60	0.941	0.006	**1.000**	**1.000**	**1.000**	0.111	0.178	0.009	—

Tabela 3.18: Wyniki testu uciąglania danych *DNA*

cech. Dzięki temu metoda ta jest zdecydowanie bardziej efektywna od pozostałych, ponieważ proces uczenia modeli adaptacyjnych na ogół trwa zdecydowanie krócej niż w przypadku przestrzeni o kilkakrotnie większych liczbach cech.

Warto zauważyć, że zwykle, różnice dokładności klasyfikacji pomiędzy modelami z różnymi sposobami uciąglania są bardzo małe w porównaniu z różnicami liczonymi względem modeli nie stosujących uciąglania.

3.8 Selekcja cech

W systemach drzew decyzyjnych selekcja istotnych dla klasyfikacji cech jest integralną częścią podstawowego algorytmu. Inne systemy (zwłaszcza te posługujące się odległościami w przestrzeni) są bardzo wrażliwe na zmiany przestrzeni klasyfikacji polegające na odrzucaniu bądź dodawaniu cech. Zdolność określania istotności cech, jaką posiadają drzewa decyzji SSV, może być wykorzystana także w innych systemach.

Problem oceny przydatności różnych cech dla klasyfikacji rozpatruje się w dwóch kategoriach: *rankingu* oraz *selekcji* cech. Przedstawienie *rankingu* cech polega zwykle na oszacowaniu zdolności predykcyjnych każdej z cech z osobna i uporządkowaniu cech w kolejności malejących wartości ocen. W praktyce najczęściej szukamy zestawu cech, który pozwoli systemom wykorzystującym odległości między wektorami budować najskuteczniejsze modele w możliwie krótkim czasie. Uporządkowanie cech, którym jest ranking, nie zawsze przynosi tu dobre wyniki, ponieważ niezależna ocena każdej z cech nie musi być zgodna z oceną grup cech – cecha druga w rankingu może nieść ze sobą niemal tę samą informację, co pierwsza, i wówczas przestrzeń tych dwóch cech może nie dawać większych możliwości niż pojedyncza cecha, z kolei cecha, która jest umieszczona na dalszej pozycji rankingu może być źródłem informacji dobrze uzupełniającej informację pierwszej cechy, więc wybór tych dwóch cech będzie tu zdecydowanie trafniejszy.

Ocena różnych kombinacji cech i hierarchiczny wybór cech z uwzględnieniem tych, które zostały wybrane w poprzednich etapach, nazywa się *selekcją* cech. Zwykle jest ona zdecydowanie bardziej złożona obliczeniowo, jednak drzewa decyzji, dzięki swej hierarchicznej strukturze mogą udostępniać pożądaną informację bez konieczności wykonywania prawie żadnych dodatkowych obliczeń.

Metody selekcji cech dzieli się na *filtry* (ang. *filters*) i *nakładki*[4] (ang. *wrappers*) [110]. *Filtry* to metody, które oceniają przydatność poszczególnych cech niezależnie od modelu, który docelowo ma być używany. Wyznaczają one grupę

[4]Słowo *nakładka* zostało zaproponowane jako zdecydowanie lepiej opisujące logikę wykonywanego zadania niż polskie tłumaczenia angielskiego *wrapper*.

najistotniejszych cech, która potem może być wykorzystana do uczenia systemów klasyfikacji. *Nakładki* to metody, które są stosowane w powiązaniu z konkretnymi systemami klasyfikacji i wykorzystują te systemy w celu doboru optymalnego zestawu cech. Taka metodologia wymaga wykonania wielokrotnych procesów adaptacyjnych, co w praktyce może być bardzo kosztowne. W związku z tym, analiza wszystkich możliwych podzbiorów cech jest zwykle niemożliwa (potrzeba selekcji cech jest największa w przypadku bardzo dużej ich liczby), co sprawia, że w praktycznych zastosowaniach możliwe jest uzyskanie lepszych wyników dzięki zastosowaniu filtrów niż z użyciem nakładek [48].

Wiele filtrów stworzono na bazie teorii informacji. Do oceny ilości informacji wzajemnej (ang. *mutual information*) pomiędzy wartościami danej cechy a klasami obiektów, koniecznym jest by badana cecha była dyskretna. Zastosowanie metody opartej na SSV do dyskretyzacji cech ciągłych (rozdział 3.6), pozwala uzyskać rzetelniejszą ocenę informatywności cech.

Filtr oparty na SSV opisuje następujący algorytm:

Algorytm 3.10 (Filtr selekcji cech oparty na kryterium SSV)

► ***Dane:*** *Przestrzeń X, zbiór klas C, zbiór treningowy $T \subseteq X \times C$, parametry metody budowania drzew* SSV.

◄ ***Wynik:*** *Lista cech uporządkowana w kolejności malejących wartości oceny ich przydatności.*

1. *Budujemy drzewo* SSV *(nazwijmy je D) zgodnie z zadanymi parametrami.*
2. *Dla każdego węzła W, który nie jest liściem drzewa wyznaczamy wartość $G(W) = E(W) - E(W_1) - E(W_2)$, gdzie W_1 oraz W_2 są podwęzłami W, a $E(X)$ oznacza liczbę wektorów ze zbioru T, które wpadają do węzła X, ale należą do innej klasy niż etykieta węzła X.*
3. *Niech $\mathscr{F}$ będzie zbiorem wszystkich cech przestrzeni X.*
4. *$i \leftarrow 0$*
5. *Powtarzamy:*
   (a) *Przypisujemy każdej cesze $F \in \mathscr{F}$ nie wykorzystanej w drzewie D rangę $R(F) = i$. Usuwamy te cechy ze zbioru $\mathscr{F}$.*
   (b) *Usuwamy z drzewa D podwęzły wszystkich węzłów W będących rodzicami liści dla których $G(W)$ przyjmuje wartość minimalną.*
   (c) *Przycinamy drzewo D do stopnia* 0.

(d) $i \leftarrow i+1$

tak długo jak $\mathscr{F} \neq \emptyset$.

6. *Wynikiem algorytmu jest lista cech uporządkowana według malejących wartości* $R(F)$.

3.8.1 Rezultaty

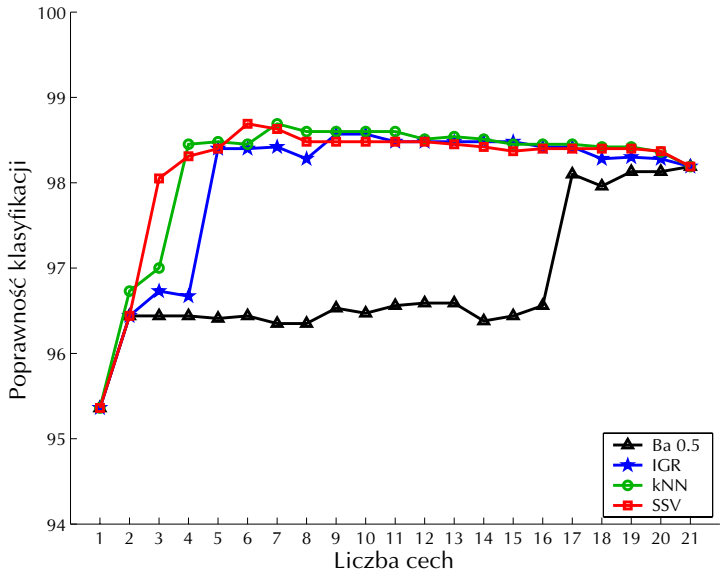
Duch i in. w [49] podjęli próbę porównania kilku metod selekcji cech. Między innymi testowano filtr zdefiniowany algorytmem 3.10 oraz nakładkę na system kNN, a także dwie metody oparte na teorii informacji. Analizie poddano zbiór *Hypothyroid* z repozytorium UCI.

Tabela 3.19 przedstawia listy cech uporządkowane według ważności (od lewej do prawej) wyznaczonej użytymi metodami.

Metoda	Numery cech od najważniejszej do najmniej ważnej																			
kNN	17	3	8	19	21	5	15	7	13	20	12	4	6	9	10	18	16	14	11	1	2
BA b=0.5	17	21	15	13	7	9	5	12	8	6	4	16	14	10	11	2	3	1	18	20	19
IGR	17	21	19	18	3	7	13	10	8	15	6	16	5	20	4	1	12	2	11	9	14
SSV BFS	17	21	3	19	18	8	1	20	12	13	15	16	14	11	10	9	7	6	5	4	2
SSV beam	17	8	21	3	19	7	9	11	10	12	14	15	16	18	20	13	6	5	4	2	1

Tabela 3.19: Porządek cech wynikający z różnych metod selekcji

Dla kolejnych zestawów cech wynikających z tej kolejności (a więc najpierw dla przestrzeni jednowymiarowej będącej najważniejszą cechą, potem dwuwymiarowej złożonej z dwóch pierwszych cech z listy itd.), uruchomiono procesy adaptacji parametrów modelu SVM i sprawdzono poprawność klasyfikacji nauczonego modelu na zbiorze testowym. Wyniki poszczególnych modeli SVM przedstawia rysunek 3.8. Linie wykresu odpowiadają metodom selekcji cech. Wykres ten dowodzi, że kryterium SSV może być bardzo skutecznym narzędziem selekcji cech. Już przy selekcji trzech cech pozwala systemowi SVM uzyskać całkiem wysoką poprawność klasyfikacji (zdecydowanie wyższą niż przy użyciu innych metod selekcji), a przy wyborze sześciu daje maksymalną dokładność testową jaką udało się uzyskać (nakładka na kNN pozwoliła uzyskać identyczną dokładność ale dla naboru siedmiu cech).



Rysunek 3.8: Porównanie metod selekcji cech na podstawie wyników klasyfikacji na zbiorze testowym uzyskanych systemem SVM

3.9 Problem niepełności danych

W rozdziale 2.14 przedstawiono sugestię dotyczącą postępowania systemów opartych na odległościach w obliczu niepełności danych. Podobny punkt wyjścia należy obrać także w przypadku drzew decyzji – jeśli nie są znane wartości danej cechy dla wszystkich wektorów treningowych, to wartość SSV jest pewną zmienną losową, której wartość oczekiwaną należy wyznaczyć.

Zauważmy, że moc zbioru T przestrzeni X można określić probabilistycznie jako

$$|T| = \sum_{x \in X} P(x \in T). \tag{3.16}$$

W związku z tym, określenie 3.3 kryterium SSV może przybrać postać:

$$\mathsf{SSV}(s,F,D) = 2 \cdot \sum_{c \in C} \sum_{x,y \in D} P(x \in \mathsf{LS}(s,F,D_c)) \cdot P(y \in \mathsf{RS}(s,F,D \setminus D_c))$$

$$- \sum_{c \in C} \min\Big(\sum_{x \in D} P(x \in \mathsf{LS}(s,F,D_c)), \sum_{x \in D} P(x \in \mathsf{RS}(s,F,D_c))\Big) \tag{3.17}$$

Dla wektorów, dla których nie znamy wartości cechy F, prawdopodobieństwa występujące w tym wzorze można oszacować na podstawie rozkładu znanych wartości cechy F w zbiorze treningowym.

W przypadku klasyfikacji wektora przez nauczone drzewo wystarczy analizować prawdopodobieństwa wpadania do kolejnych węzłów drzewa, a następnie zsumować dla każdej z klas prawdopodobieństwa przynależenia do niej wynikające z przynależności do odpowiednich liści drzewa. Ostateczna decyzja klasyfikatora będzie odpowiadała maksymalnemu prawdopodobieństwu

3.10 Drzewa jako klasyfikatory probabilistyczne

Możliwość opisu klasyfikacji drzewa poprzez reguły logiki klasycznej jest zwykle bardzo pożądaną cechą, która sprawia, że drzewo decyzji bywa preferowanym rozwiązaniem nawet wówczas, gdy istnieją inne systemy, które potrafią dokładniej klasyfikować dane. Ta ogromna zaleta drzew jest jednak zarazem przyczyną najczęściej kierowanych pod ich adresem zarzutów, które to dotyczą „ostrości" podejmowanej decyzji. Klasyczne drzewa nie stopniują decyzji tzn. nie opisują klasyfikacji prawdopodobieństwem, które by obrazowało łagodne przejście z jednej klasy do drugiej. Dwa wektory, które leżą bardzo blisko siebie mogą być klasyfikowane do różnych klas. Nie różnicuje się też decyzji dla przypadków bardzo typowych i tych leżących blisko granic decyzji. Ostre decyzje są interpretowane jako prawdopodobieństwa równe 0 bądź 1, co sprawia, że np. w analizie błędów aproksymacji czy krzywych ROC, takie klasyfikatory nie dają satysfakcjonujących rezultatów.

Najprostszym ze stosowanych rozwiązań mających na celu „złagodzenie" decyzji drzew klasyfikacji jest przypisywanie wektorowi prawdopodobieństw przynależenia do poszczególnych klas odpowiadających rozkładowi klas w podzbiorze wektorów treningowych wpadających do tego samego liścia co badany wektor. Jeśli więc wektor wpada do liścia, do którego należy n wektorów treningowych, spośród których n_1 należy do klasy C_1, ..., n_k do C_k (załóżmy że zbiór klas ma k elementów), to prawdopodobieństwo przynależenia do klasy C_k będzie oszacowane jako $P(C(x) = C_k) = \frac{n_k}{n}$.

Takie rozwiązanie ma jednak szereg poważnych wad, bo np. liściom, w które wpada bardzo mało wektorów treningowych, przypisuje się prawdopodobieństwa bliskie 0 bądź 1, choć tak naprawdę decyzja nie jest aż tak bardzo wiarygodna. Mocno obcięte drzewa pozwolą uniknąć takich problemów, ale za to mogą dawać kiepskie dokładności klasyfikacji.

Provost i Domingos w [148] zaproponowali, by w powyższym oszacowaniu prawdopodobieństw nanieść poprawki zwane *korektą Laplace'a*. Dla danego wek-

tora, który wpada do węzła W szacują prawdopodobieństwa:

$$P(C(x) = C_i) = \frac{N_i^W + \lambda_i}{N^W + \sum_{j=1}^{k} \lambda_j}, \tag{3.18}$$

gdzie N^W jest liczbą wektorów zbioru treningowego należących do W, N_i^W jest liczbą wektorów zbioru treningowego należących do W i reprezentujących klasę C_i oraz λ_i są stałymi ustalonymi dla poszczególnych klas. W testach używali wszystkich λ_i równych 1, co daje prostszą postać:

$$P(C(x) = C_i) = \frac{N_i^W + 1}{N^W + k}. \tag{3.19}$$

Celem tej korekty jest więc zbliżenie prawdopodobieństw do $\frac{1}{k}$ dla małych liści.

Dodatkowo Provost i Domingos sugerują użycie baggingu i uśrednienie prawdopodobieństw po wszystkich jego przebiegach. Twierdzą także, że korzystne jest budowanie bardzo szczegółowych drzew, więc korzystając z drzew C4.5 wyłączają wszelkie opcje upraszczania, by analizie poddać maksymalne drzewo. Dzięki zróżnicowaniu w ten sposób prawdopodobieństw, można oceniać ich jakość na podstawie krzywych ROC i analizy pola pod krzywą (AUC). Podobnie jak w przypadku nadmiernie obcinanych drzew tak i tutaj przy nadmiernym ich rozbudowywaniu trzeba mieć na względzie, że choć wpływa to korzystnie na pole pod krzywą ROC, i zwiększa dokładność liczoną na zbiorze treningowym, to jednak zmniejsza przy tym poprawność na nie widzianych danych.

Innym rozwiązaniem mającym na celu wygładzenie decyzji drzewa jest stosowana również do klasyfikatorów Bayesowskich [111] korekta Laplace'a z parametrem m [29], zwana też *m-oszacowaniem*, wg której

$$P(C(x) = C_i) = \frac{N_i^W + b \cdot m}{N^W + m}, \tag{3.20}$$

gdzie b jest poprawnością podstawową a m parametrem metody.

Jeszcze inne podejście proponują Zadrozny i Elkan [187]: stosowne prawdopodobieństwa liczy się nie dla finalnego węzła, ale dla ostatniego węzła gałęzi, do którego należy przynajmniej v wektorów treningowych (v to parametr metody).

Buntine przedstawił [28] podobną ideę, która liczy prawdopodobieństwa jako ważoną sumę częstości dla różnych węzłów gałęzi.

Margineantu i Dietterich opracowali [123] metodę wykorzystującą *drzewa z opcjami* (ang. *option trees*), które w danym węźle mogą sprawdzać kilka alternatywnych warunków, dzięki czemu dokładniej odzwierciedlają prawdopodobieństwa. Niestety w tym przypadku struktura drzewa nie jest tak czytelna jak dla drzew w klasycznej postaci.

Wszystkie powyższe rozwiązania, mimo lepszej oceny prawdopodobieństw, mają tę wadę, że oceniają je dla pewnego obszaru (wynikającego ze struktury drzewa), a nie dla konkretnego przypadku. Nie mają one zdolności rozróżniania prawdopodobieństw dla punktów leżących blisko granicy decyzyjnych i punktów znajdujących się z dala od granic, jeśli tylko wpadają do tego samego liścia.

Podobnie jak można uzyskać prawdopodobieństwa spełniania reguł poprzez założenie pewnego rozmycia danych (rozdział 5.2), tak i na podstawie nauczonego drzewa można generować klasyfikatory probabilistyczne. Wystarczy odpowiednio dobrać parametry rozmyć określające niepewność poszczególnych wymiarów. W najprostszej wersji można uznać, że rozmycie (w sensie odchylenia standardowego rozkładu normalnego) w każdym wymiarze jest częścią zakresu możliwych wartości (tą samą częścią dla każdego wymiaru) i wówczas wystarczy dobrać jeden parametr, co nie jest obliczeniowo zbyt kosztowne.

Tę samą ideę można zrealizować oszacowaniami prawdopodobieństw metodą *całkowania Monte Carlo*. Losowanie punktów w okolicy badanego wektora z uwzględnieniem odpowiednich rozkładów pozwala uniknąć założenia niezależności cech przestrzeni, ale dla uzyskania dokładnych oszacowań wymaga generowania dużej liczby punktów w wielowymiarowych przestrzeniach.

Również niezbyt kosztowne powinno być zastąpienie ostrych cięć realizowany przez warunki logiczne funkcjami sigmoidalnymi o przesunięciach (ang. *bias*) odpowiadających punktom cięcia oraz odpowiednio dobranych w procesie optymalizacji nachyleń (ang. *slope*).

Zadania te nie zostały jeszcze zrealizowane – mogą być przedmiotem dalszych badań.

3.11 Możliwości dalszych badań

Kryterium SSV znalazło zastosowanie w rozwiązywaniu wielu problemów dotyczących klasyfikacji danych. Ciągle jednak pozostaje wiele możliwości nowych zastosować kryterium oraz udoskonalania dotychczas opracowanych na jego bazie metod.

Wykorzystanie lasów. Możliwość konstruowania wielu alternatywnych drzew jest bardzo atrakcyjna, kiedy szukamy akceptowalnego dla eksperta wyjaśnienia klasyfikacji obiektów. Interesujące mogą być wyniki złożonych systemów klasyfikacji zbudowanych na bazie drzew dostępnych w lesie. Dla uzyskania w pełni automatycznej metody tworzącej takie komitety, potrzebne są metody pozwalające wybrać z lasu drzewa wartościowe z punktu widzenia komitetu. Oczekiwane

rezultaty, to systemy o możliwościach zbliżonych do tych, które można uzyskać przez zastosowanie metod wielokrotnego próbkowania (p. rozdział 2.13, jednak uzyskane dużo mniejszym nakładem mocy obliczeniowych, bo do wygenerowania lasu wystarczy pojedynczy proces adaptacji.

Klasyfikatory probabilistyczne. Dla wzbogacenia możliwości interpretacyjnych klasyfikatorów opartych na SSV warto przeanalizować inne (niż najbardziej podstawowe) metody określania prawdopodobieństw przynależności do poszczególnych klas. Szczególnie wartościowe będą tutaj sygnalizowane w rozdziale 3.10 metody lokalne tj. określające prawdopodobieństwa dla zadanego wektora a nie dla całego obszaru (wynikającego z granic klasyfikacji drzewa decyzji), w który ten wektor wpada.

Metody szukania drzew. W związku z wnioskami o małej atrakcyjności wyników uzyskiwanych dokładnymi metodami szukania, warto opracować metody mieszane, które przy stosunkowo małej złożoności będą dawały ciekawe rezultaty. Jedną z możliwości jest wykonanie standardowego procesu szukania metodą „najpierw najlepszy", dla określenia złożoności optymalnego z punktu widzenia generalizacji drzewa, a następnie zastosowanie metod dokładniejszego szukania w celu znalezienia modeli o nie mniejszej dokładności, a jednocześnie nie większej złożoności. W myśl zasady minimalnej długości opisu, taka metoda nie powinna wpływać negatywnie na zdolności generalizacyjne modelu, więc daje duże szanse znajdowania jeszcze prostszych opisów o wyższej dokładności klasyfikacji.

Uczenie na poziomie *meta*. Bardzo szybko rozwijającą się gałęzią Inteligencji Obliczeniowej stały się ostatnio systemy działające na poziomie *meta*, tzn. takie, które zajmują się doborem optymalnych algorytmów adaptacyjnych oraz ich parametrów. Często dobór parametrów uczenia jest zadaniem bardzo pracochłonnym. Można przy użyciu danego systemu zbudować bardzo dobry model, ale trzeba „odgadnąć" jakich parametrów należałoby użyć.

Również metody indukcji drzew decyzji mają pewne parametry, od których zależy otrzymany wynik. Zastosowanie odpowiednich metod analizy i szukania może pozwolić w sposób automatyczny (choć oczywiście wymagający zdecydowanie większego nakładu czasu) odnajdywać modele bliskie optymalnym.

Systemy klasyfikacji nie wymagające dostrajania żadnych parametrów będą w przyszłości na pewno bardzo pożądane, bo pozwolą na szerokie zastosowania w przemyśle, bez konieczności posiadania wiedzy ekspertów przez użytkowników korzystających z tych metod.

Ciągłe uogólnienie kryterium. Kryterium SSV jest funkcją kawałkami stałą dla parametrów określających punkty podziału. Jej nieciągłość zmusza do korzystania z metod szukania w celu znajdowania ekstremów.

Założenie nieprecyzyjności danych bądź punktów podziału pozwala zdefiniować kryterium, które w granicy rozmyć równej zeru jest tożsame z kryterium SSV, natomiast jest funkcją ciągłą. Wystarczy przyjąć następującą definicję:

$$\mathsf{SSV}^c(s,F,D) = \sum_{c\in C}\sum_{x,y\in D} P(x \in \mathsf{LS}(s,F,D_c)) \cdot P(y \in \mathsf{RS}(s,F,D \setminus D_c)) \qquad (3.21)$$

Drugi człon kryterium SSV może w tym przypadku zostać pominięty, bo uciąglenie funkcji spełnia dodatkowo podobną rolę. Przy założeniu gaussowskich niepewności danych bądź punktów podziału można (podobnie jak w rozdziale 5.2) z łatwością wyznaczać wymagane prawdopodobieństwa, a dzięki różniczkowalności funkcji mamy możliwość korzystania z gradientowych metod minimalizacji.

Interesującą byłaby także analiza wyników gradientowych metod optymalizacji, w których rozmycia danych byłyby również parametrami adaptacyjnymi.

Agregacja cech. Kryterium separowalności będące funkcją różniczkowalną pozwoliłoby także na łatwe tworzenie nowych cech, które skupiałyby w sobie informację rozproszoną w wielu oryginalnych cechach. Można na przykład maksymalizować wartość ciągłej funkcji separowalności dla rzutów wektorów treningowych na prostą, wyznaczając w ten sposób wartościową kombinację liniową dostępnych w bazie cech obiektów. Można również niedużym kosztem szukać interesujących nieliniowych transformacji cech.

Rozdział 4

Sieci neuronowe dla indukcji reguł

Granice decyzji sieci neuronowych są zwykle nieliniowe, więc klasyfikatory te nie mają prostego opisu logicznego. Można jednak przy nałożeniu pewnych ograniczeń na sieci typu MLP budować struktury, które mają naturalną interpretację logiczną.

Dwa pokrewne algorytmy, przedstawione w tym rozdziale, są modyfikacjami metody wstecznej propagacji błędu, pozwalającymi na przekształcenie wielowarstwowych perceptronów (MLP) w sieci, których działanie można łatwo zinterpretować w postaci formuł logicznych (*logical networks* – LN). Stąd pochodzą nazwy tych metod, **MLP2LN** i w wersji konstruktywistycznej **C-MLP2LN** [43, 44, 75, 45]. Łączą one zalety sieci neuronowych (możliwość uczenia metodami gradientowymi) oraz systemów logicznych (zrozumiała forma budowanych modeli).

Interesujące wyniki można uzyskać także stosując powszechnie znane metody szukania w celu znalezienia optymalnej sieci MLP. Heurystyczne procesy szukania w odpowiednio ograniczonym obszarze potrafią budować modele równie szybko jak gradientowe algorytmy minimalizacji, a dodatkowo mogą konstruować modele dające się opisać językiem logiki. Takim procesem jest metoda S-MLP (ang. *Search-based MLP*) [51].

4.1 MLP2LN i C-MLP2LN

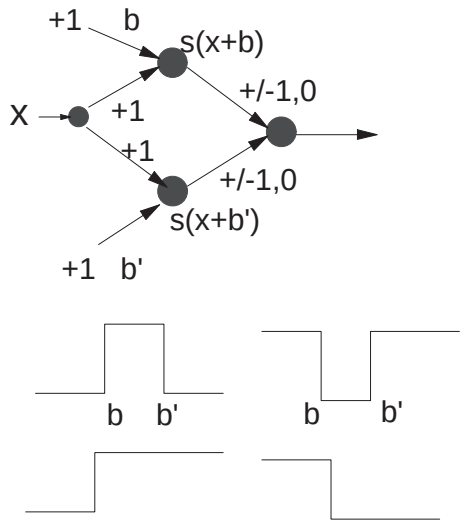
Aby sieć typu MLP mogła zostać przetransformowana do postaci reguł logiki klasycznej, wykorzystywane funkcje aktywacji muszą powodować maksymalne wzbudzenie neuronów albo całkowity brak wzbudzenia. Dlatego neurony, z których zbudowane są sieci MLP2LN i C-MLP2LN realizują funkcje sigmoidalne wzbogacone o parametr s (zwany skosem), którym można zmieniać stromość funkcji tak by w granicy przy s zmierzającym do nieskończoności funkcje te były

równoważne funkcjom progowym:

$$f(x) = \frac{1}{1+e^{-s(Wx+b)}}. \tag{4.1}$$

Zatem funkcje te łączą w sobie dwie zalety: dla dużych wartości skosu dają się wprost przekładać na język logiki, a mniejsze skosy pozwalają efektywnie wykorzystywać metody gradientowe do trenowania sieci.

Kontekstowe zmienne lingwistyczne. Lingwistyczne jednostki neuronów (nazywane jednostkami *L*) automatycznie analizują wejścia i produkują zmienne lingwistyczne [44]. Pomysł oparty jest na „funkcjach okienkowych", które można uzyskać z kombinacji dwóch neuronów z funkcjami sigmoidalnymi o różnych wartościach progów b i b'. Różnice dwóch sigmoid reprezentują typową zmienną lingwistyczną równoważną warunkowi $x \in [b, b']$ lub jego zaprzeczeniu. Sumy podobnie jak pojedyncze sigmoidy realizują przedziały jednostronnie nieskończone. Wartości progów są parametrami sieci, które podlegają procesowi adaptacji. Wszystkie sigmoidy w końcowym etapie uczenia stają się bardzo strome, dzięki czemu wiernie reprezentują przedziały.



Rysunek 4.1: Schemat jednostki *L*

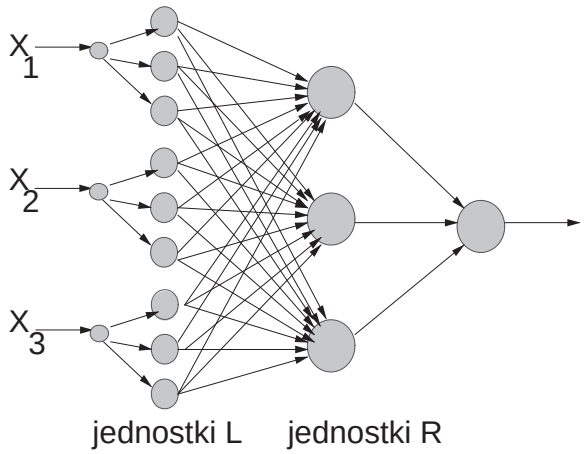
Schemat jednostki *L* jest pokazany na rysunku 4.1. Składa się ona z neuronu wejściowego, który jest połączony wagami ustawionymi na 1 i nie podlegającymi

uczeniu z dwoma neuronami wewnętrznymi, które z kolei połączone są z pojedynczym neuronem wyjściowym jednostki. Wagi dochodzące do neuronu wyjściowego mogą przybierać wartości 0, $+1$ lub -1, co daje możliwość realizacji przedziałów skończonych, a także lewostronnie i prawostronnie nieskończonych.

Można oczywiście zamiast jednostek L użyć metod dyskretyzacji danych tak, by sieć nie dostawała na wejściu sygnałów ciągłych, jednakże rozbudowanie sieci o jednostki L daje możliwość realizacji celu szukania zmiennych lingwistycznych w kontekście powstających reguł, a co za tym idzie większe szanse na zwarte i skuteczne reguły. Dodatkowo użycie jednostek L sprowadza etapy wyboru zmiennych lingwistycznych i tworzenia reguł do jednego zadania.

Kiedy używamy zdyskretyzowanych danych pojedyncze wejście dla cechy o ciągłych wartościach musi zostać zamienione na wektor elementów wejściowych, składających się z wartości ± 1. Wektor ten ma wymiar równy liczbie możliwych (dyskretnych) wartości przyjmowanych przez daną cechę wejściową. Na przykład cecha, która może przyjmować trzy wartości lingwistyczne: *mały*, *średni* i *duży*, musi zostać wprowadzona do sieci jako trzy niezależne wejścia odpowiadające tym wartościom lingwistycznym. Jeśli cecha przyjmuje wartość *mały*, to na pierwszym z tych trzech wejść postawimy wartość $+1$ a na pozostałych -1, co pozwoli łatwo odnajdywać także reguły z negacjami. A zatem trzy wartości lingwistyczne: mały, średni i duży, zostaną jako wektory wejściowe zakodowane w następujący sposób: mały=$[+1,-1,-1]$, średni=$[-1,+1,-1]$ oraz duży=$[-1,-1,+1]$.

Struktura sieci. Sieć MLP2LN składa się z trzech warstw: wejściowej, ukrytej i wyjściowej (kiedy używamy jednostek L mamy w rzeczywistości większą liczbę warstw, ale dla uproszczenia opisu traktujemy je jako część warstwy wejściowej). Liczba węzłów w warstwie wyjściowej równa jest liczbie klas w zbiorze treningowym, natomiast w warstwie wejściowej, w zależności od tego, czy korzystamy z jednostek L czy dyskretyzacji, liczbie cech przestrzeni obiektów bądź liczbie zmiennych lingwistycznych. Każdy z neuronów w warstwie ukrytej jest połączony ze wszystkimi węzłami z warstwy wejściowej i z jednym neuronem wyjściowym (będzie realizował reguły klasyfikujące do klasy odpowiadającej temu wyjściu). Schemat części sieci odpowiadającej jednej z klas przedstawia rysunek 4.2. Początkowa liczba węzłów w warstwie ukrytej jest zależna od tego czy stosujemy standardową wersję algorytmu czy konstruktywistyczną (w podejściu konstruktywistycznym dodajemy kolejne neurony w tracie uczenia pojedynczo, w miarę potrzeby, natomiast w podejściu standardowym od początku do końca uczenia mamy tę samą strukturę z określoną na początku liczbą neuronów). Podczas procesu uczenia wymuszane są wartości wag równe 0, $+1$ lub -1. Analizując wagi i próg dla neuronu z warstwy ukrytej otrzymujemy reguły odnoszące się do klasy, z którą ten neuron jest połączony (połączenie z określonym węzłem wyjściowym).



Rysunek 4.2: Struktura sieci MLP2LN

Jeśli waga połączenia jest równa $+1$ to otrzymujemy reguły dla danej klasy, jeśli natomiast waga jest równa -1 to wyjątki, czyli reguły opisujące przypadki błędnie klasyfikowane przez istniejące reguły dla tej klasy. Węzły w warstwie wyjściowej dokonują jedynie sumowania aktywacji odpowiednich węzłów z warstwy ukrytej, więc w związku z tym, że na wyjściu oczekujemy wartości 0 lub 1, sytuacja, gdy dwa węzły klasyfikują ten sam wektor traktowana jest jako błąd. Dzięki temu otrzymujemy z różnych węzłów reguły, które są rozłączne, czyli nie klasyfikują tych samych wektorów.

Algorytm uczenia. Logiczna interpretacja węzłów w sieci MLP jest w ogólności trudna, dlatego algorytm MLP2LN używa funkcji sigmoidalnych, których nachylenia w czasie uczenia stopniowo wzrastają. Proces adaptacji wymusza wartości wag równe $0,+1,-1$. Wartość 0 oznacza że zmienna wejściowa połączona tą wagą jest nieistotna, $+1$ oznacza, że dana wartość cechy będzie sprzyjać aktywacji neuronu, oraz -1, że wartość ta będzie zmniejszać aktywację. To wymuszanie odpowiednich wartości wag można osiągnąć poprzez modyfikacje funkcji błędu stosowanej dla algorytmu wstecznej propagacji:

$$E(W) = \frac{1}{2}\sum_p \sum_k \left(Y_k^{(p)} - \mathbf{A}_W\left(X^{(p)}\right)_k\right)^2 + \frac{\lambda_1}{2}\sum_{i>j} W_{ij}^2 + \frac{\lambda_2}{2}\sum_{i>j} W_{ij}^2 (W_{ij}-1)^2 (W_{ij}+1)^2 \tag{4.2}$$

Można zastosować również człony kary w innej postaci [45], np. prostszym członem trzeciego stopnia:

$$|W_{ij}||W_{ij}^2 - 1| \tag{4.3}$$

Nową postać przyjmuje również gradient, mamy więc dodatkowe człony we wzorze na zmianę parametrów sieci. Dla funkcji błędu zdefiniowanej wzorem 4.2 dodatkowe człony gradientu to:

$$\lambda_1 W_{ij} + \lambda_2 W_{ij}(W_{ij}^2 - 1)(3W_{ij}^2 - 1). \tag{4.4}$$

Pierwszy składnik wymusza podczas uczenia małe wartości wag przez co prowadzi do eliminacji cech zbędnych, natomiast drugi wymusza dla wag wartości $-1{,}0$ lub $+1$, umożliwiając w ten sposób późniejszą logiczną interpretację sieci. Za pomocą parametrów λ_1, λ_2 możemy zwiększać lub też zmniejszać dominację odpowiednich członów. Ustalenie dominacji któregoś z członów wyznacza granicę miedzy prostotą a dokładnością reguł otrzymanych z sieci. Jeżeli chcemy uzyskać bardzo prostą sieć, a co za tym idzie proste reguły dające przybliżony opis danych, to parametr pierwszego członu powinien być tak duży, jak to tylko jest możliwe, przy akceptowalnym jeszcze błędzie.

Na początku procesu uczenia parametr λ_2 jest równy zero natomiast λ_1 jest mały (ma np. wartość 0.00001). Z takimi parametrami uczymy sieć tak długo, jak długo maleje błąd. Następnie zwiększamy wartość parametru λ_1 (np. do 0.10) i kontynuujemy uczenie. Z reguły po zwiększeniu parametru λ_1 następuje wzrost błędu SSE (ang. *Summed Squared Error* – suma kwadratów błędów) – można go nieco zmniejszyć przez zwiększenie skosów.

Tę procedurę uczenia powtarzamy tak długo, aż zaobserwujemy, że większość wag ma wartość zero lub też nastąpił bardzo duży skok błędu. W tym momencie usuwamy zbędne połączenia, człon odpowiedzialny za wymuszanie małych wartości wag zostaje wyłączony przez ustawienie λ_1 na 0, natomiast człon drugi uaktywniony. Parametr λ_2 przyjmuje wartość równą lub też trochę większą od ostatniej wartości parametru λ_1. Jednocześnie nadal zwiększamy nachylenie funkcji sigmoidalnych, realizowanych przez węzły sieci. W celu dalszego zmniejszenia wartości wag można jednocześnie z niezerowym parametrem λ_2 utrzymywać niezerową wartość parametru λ_1. Wówczas wartość parametru λ_1, początkowo istotnie większa od λ_2, w miarę zwiększania λ_2 stopniowo maleje do zera.

W przypadku trudniejszych danych warto spróbować kilku strategii zmiany parametrów po to, by uzyskać jak najprostsze reguły. Kontynuujemy proces uczenia zwiększając wartość parametru λ_2 oraz nachylenia sigmoid. Wraz ze wzrostem wartości parametru λ_2 wagi stają się coraz bliższe docelowych wartości. Parametr λ_2 nie powinien przekraczać wartości 1. Jeśli parametr ten osiągnął już swoją maksymalną wartość a parametr uczenia, powoli zmniejszany w procesie uczenia, nie jest jeszcze bardzo mały, tzn. jest > 0.00001, to zwiększamy skos

sigmoid i zmniejszamy parametr uczenia, aż osiągnie wartość minimalną (bliską bądź wręcz równą 0).

W końcowym etapie uczenia zwiększamy nachylenie sigmoid do bardzo dużych wartości (np. 1000), przez co uzyskujemy ostre granice decyzyjne.

Początkowe wymuszenie małych wartości wag umożliwia w późniejszym etapie wyzerowanie tych wag zupełnie, natomiast pozostałe wagi będą zbliżały swoją wartość do ± 1 dzięki drugiemu członowi. Może się zdarzyć, że na skutek zbyt mocnego wymuszenia wag o małych wartościach w pierwszej fazie, w końcowym etapie uzyskuje się węzeł, który posiada wszystkie wagi zerowe. W takim przypadku trzeba ponownie węzeł zainicjować i powtórzyć proces uczenia utrzymując mniejszą wartość parametru λ_1. Nawet jeśli z nauczonego węzła ukrytego otrzymujemy proste reguły, dobrze jest spróbować ponownie nauczyć sieć, jeszcze bardziej wymuszając, w początkowym etapie, zerowe wagi. Taka próba może doprowadzić do jeszcze prostszej postaci sieci, a co za tym idzie i reguł, ponieważ liczba reguł, która zostanie utworzona z danego węzła bardzo mocno zależy od liczby niezerowych wag. Dlatego też, pierwszy etap uczenia (wymuszanie małych wag) jest bardzo istotny dla efektu generowania reguł. Cała procedura wymuszania odpowiednich wartości parametrów sieci dotyczy tylko i wyłącznie wag – wszystkie progi mogą przyjmować dowolne wartości.

Mimo, że dodatkowe człony w funkcji błędu nie zmieniają MLP dokładnie w sieć logiczną, to ułatwiają w znaczny sposób logiczną interpretację końcowej sieci.

Interpretacja węzłów ukrytych. Na koniec procesu adaptacyjnego, wszystkie sygnały wejściowe oraz wagi pomiędzy warstwą wejściową a wyjściową mają wartości $+1,-1$ lub 0, dlatego też sygnał wpływający do węzła ukrytego ma wartości całkowite. Ponieważ sigmoidy w węzłach mają bardzo duży skos, to funkcja aktywacji ma wartość $+1$ bądź 0. Na podstawie analizy aktywacji oraz progu sigmoidy możemy określić regułą logiki klasycznej w jakich przypadkach węzeł może się wzbudzić. Wzbudzenie może nastąpić tylko wtedy, gdy wartość aktywacji przekroczy wartość progu. Ponieważ funkcja aktywacji ma postać 4.1, to $f(x) = 1$ gdy $e^{-s(Wx+b)} = 0$, a więc gdy $Wx+b > 0$ (przy założeniu, że skos jest bardzo duży). Żeby utworzyć reguły wystarczy analizować przypadki w których $Wx > -b$. Rozpatrzmy następujący przykład (dla prostoty analizujemy węzeł ukryty, który połączony jest tylko z jednym wejściem lingwistycznym): $W = [+1,0,0,-1]$, $b = -1$, sygnał wejściowy $x \in [x_1,x_2,x_3,x_4]$ gdzie $x_1 = [+1,-1,-1,-1]$, $x_2 = [-1,+1,-1,-1]$, $x_3 = [-1,-1,+1,-1]$, $x_4 = [-1,-1,-1,+1]$, są interpretowane jako x_1 =mały, x_2 =średni, x_3 =duży i x_4=bardzo duży. Mamy zatem: $Wx_1 = 2$, $Wx_2 = 0$, $Wx_3 = 0$ oraz $Wx_4 = -2$, czyli tylko w pierwszym przypadku $Wx > -b$, a więc reguła, która opisuje działanie takiego neuronu ukrytego ma postać:

$$x = \text{mały} \rightarrow \textit{klasa 1}.$$

Gdybyśmy natomiast mieli $b = -1$, to otrzymalibyśmy regułę

$$x = \text{mały} \vee x = \text{średni} \vee x = \text{duży} \rightarrow \textit{klasa 1},$$

co można zapisać w prostszej formie jako:

$$\neg\, x = \text{bardzo duży} \rightarrow \textit{klasa 1}.$$

Rozpatrywanie wszystkich możliwych kombinacji sygnałów wejściowych dochodzących do neuronów warstwy ukrytej przy dużej liczbie wejść staje się problemem kombinatorycznie złożonym. Można jednak znacznie uprościć proces analizując wejścia w odpowiedni sposób. Przede wszystkim należy zauważyć, że kilka wejść neuronu ukrytego może odpowiadać temu samemu wejściu sieci (wynika to z dyskretyzacji wejściowej cechy przez jednostki L), co ogranicza liczbę możliwych kombinacji sygnałów. Z drugiej strony, analiza wejść w kolejności malejących wpływów do ważonej sumy, pozwala w wielu przypadkach pominąć wiele różnych kombinacji. Na przykład kiedy jeden sygnał wejściowy ma wartość 5, a dla każdego z pozostałych sześciu wejść minimalny udział w ważonej sumie jest równy -1, oraz próg ustalił się dla analizowanego neuronu na wartość -2, to możemy zaniechać dalszej analizy, bo wiemy, że suma nie może być mniejsza niż -2 dla jakiejkolwiek kombinacji sygnałów z pozostałych wejść. Podobnie można zaniechać szukania reguły jeśli mamy pewność, że żadna z kombinacji nie pozwoli nam uzyskać sumy wyższej niż wartość progu. Taka strategia „odczytywania" reguł z sieci pozwala opisać sieć odpowiednio małą liczbą reguł klasyfikacji.

C-MLP2LN. Ponieważ liczebność zbioru reguł, które powstaną z sieci jest w dużej mierze zależna od liczby węzłów ukrytych, to problem ustalania tej liczby (poważny dla sieci typu MLP) nabiera tutaj jeszcze większego znaczenia. Problem ten można skutecznie rozwiązać stosując konstruktywistyczną wersję metody MLP2LN (stąd nazwa C-MLP2LN). Na początku tworzy się sieć, w której dla danej klasy istnieje tylko jeden neuron ukryty, który trenowany jest na wszystkich wektorach ze zbioru treningowego. Po zakończeniu uczenia dodawany jest do warstwy ukrytej nowy neuron, połączony z tą samą klasą. Poprzedni neuron zostaje natomiast „zamrożony", tzn. wagi tego neuronu podczas dalszego uczenia nie będą się zmieniały. Dzięki temu wektory wejściowe, które są poprawnie klasyfikowane przez zamrożony neuron, nie dają już wkładu do funkcji błędu. Uczymy sieć ponownie i w razie konieczności dołączamy następny neuron.

Jeśli zamrożone neurony popełniają błędy dodajemy neuron połączony z jednostką wyjściową wagą równą -1. Oznacza to, że szukamy wyjątków od działania już zbudowanego fragmentu sieci, czyli staramy się znaleźć regułę, która

opisze wektory dotychczas błędnie klasyfikowane. Całą procedurę powtarzamy tak długo, aż uzyskamy wystarczająco mały błąd, albo aż reguły, które powstają podczas analizy kolejnego nauczonego węzła staną się zbyt szczegółowe, lub będzie ich zbyt dużo. Ponieważ pierwsze węzły w sieci obejmują cały zbiór treningowy to reguły, które się z nich otrzymuje są najbardziej ogólne. Kolejne neurony dają coraz bardziej szczegółowe reguły, aż wreszcie otrzymuje się reguły opisujące pojedyncze wektory. W myśl zasady minimalnej długości opisu, takie reguły (opisujące niewielką liczbę wektorów treningowych), powinny być odrzucane, ponieważ psują generalizację. A zatem reguły generowane są w porządku od najbardziej ogólnych do coraz bardziej szczegółowych. Proces uczenia jest bardzo szybki, ponieważ w danej chwili uczony jest tylko jeden węzeł sieci.

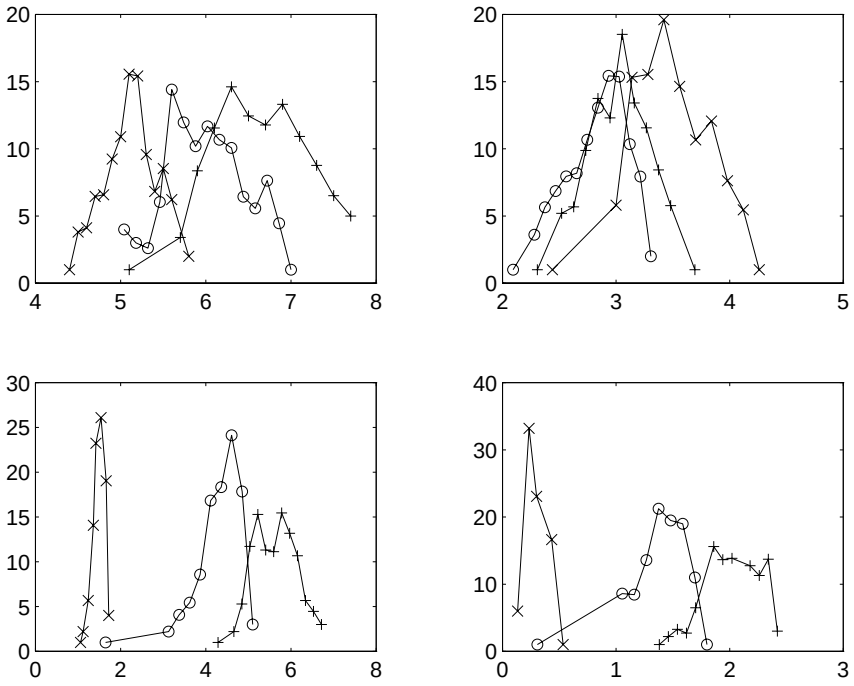
4.1.1 Rezultaty

Nasze neuronowe metody indukcji reguł logicznych zastosowaliśmy do wielu zbiorów danych, między innymi do powszechnie znanych zbiorów zawartych w bazie gromadzonej w UCI. Otrzymane wyniki pozwalają wysoko ocenić efektywność tych metod. Najczęściej wyniki plasują się w ścisłej czołówce rankingu poprawności klasyfikacji, a logiczny opis struktury danych daje możliwość zrozumienia decyzji podejmowanych przez system i w ten sposób wydobycia z danych wiedzy, która może się okazać przydatna ekspertom w danej dziedzinie. Szczególnie przydatne mogą być regułowe opisy diagnoz medycznych.

Przedstawione tutaj wyniki i porównania z osiąganymi przez inne systemy są tylko przykładami ilustrującymi działanie opisanych metod. Zestawy reguł dla większej liczby baz danych znaleźć można w pracach [43, 44, 45] oraz pod internetowym adresem *http://www.phys.uni.torun.pl/kmk/projects/rules.html*.

Iris. Dobrym przykładem ilustrującym metodologię jest przypadek danych o irysach. Ponieważ wszystkie cztery cechy opisujące dane mają charakter ciągły, to musimy albo dokonać ich dyskretyzacji przed użyciem metod MLP2LN albo też zastosować wariant sieci z jednostkami L. Najprostszą metodą uzyskania zmiennych lingwistycznych jest arbitralny podział zakresów cech na kilka równych części. Na przykład dzieląc na trzy części możemy uzyskać przedziały reprezentujące zmienne lingwistyczne o wartościach mały (s), średni (m) i duży (l). Oczywiście ostateczne wyniki będą mocno uzależnione od wybranego na początku podziału, więc stosując taką metodę dyskretyzacji mamy małe szanse uzyskać dobre reguły. W przypadku irysów podział na trzy równe części daje całkiem dobre wyniki (bo przypadkowo pasuje do optymalnego podziału dwóch najważniejszych cech, co wyraźnie widać na rysunku 4.3), ale już podział na cztery czy pięć części prowadzi do większej liczby reguł i mniejszej poprawności klasyfikacji [103].

Jeśli jednak chcemy użyć sieci MLP2LN bez jednostek L, to należy dokonać dyskretyzacji wejść stosując do tego celu specjalne metody (np. prezentowaną w rozdziale 3.6 metodę opartą na kryterium separowalności) lub histogramy, które pokazują rozkład klas w poszczególnych wymiarach. Analiza histogramów dla danych o irysach (rysunek 4.3) potwierdza, że podział na trzy równe części

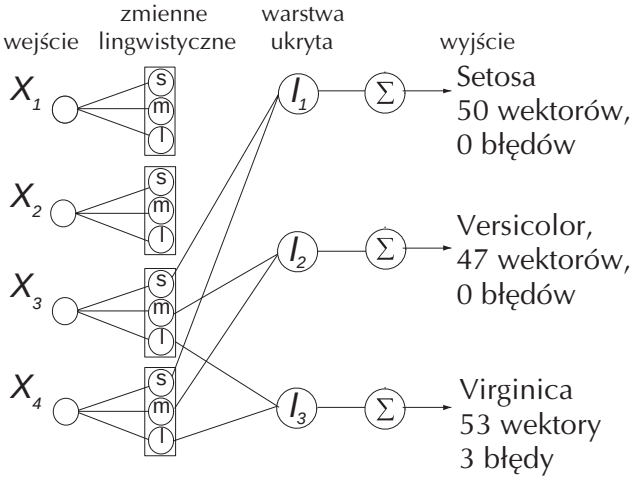


Rysunek 4.3: Histogramy dla czterech cech opisujących irysy – cechy *petal length* i *petal width* (dolne wykresy) pozwalają lepiej oddzielić różne klasy, niż pierwsze dwie

dość dobrze odzwierciedla rozkład klas. Dzieląc zakresy cech na podstawie sporządzonych histogramów otrzymujemy zmienne lingwistyczne przedstawione w tabeli 4.1. Dla takiej dyskretyzacji metoda C-MLP2LN tworzy po jednym neuronie dla każdej z klas. Struktura nauczonej sieci przedstawiona jest na rysunku 4.4. Ponieważ mamy po jednym neuronie ukrytym na klasę, a warstwa wyjściowa wykonuje proste sumowanie, to można powiedzieć, że powstała sieć nie ma warstwy ukrytej, a tylko trzy neurony w warstwie wyjściowej. Trenowanie sieci zajęło 1000 epok, a wagi połączeń w końcowej sieci odbiegają co najwyżej o 0.05 od wartości ± 1 lub od 0. Otrzymaliśmy zestaw wag i progów przedstawiony w tabeli 4.2 (dla zwiększenia przejrzystości podane są tylko znaki wag). Prosta analiza tych parametrów prowadzi do następującego zestawu reguł:

	s	*m*	*l*
sepal length	[4.3,5.5]	(5.5,6.1]	(6.1,7.9]
sepal width	[2.0,2,75]	(2.75,3.2]	(3.2,4.4]
petal length	[1.0,2.0]	(2.0,4.93]	(4.93,6.9]
petal width	[0.1,0.6]	(0.6,1.7]	(1.7,2.5]

Tabela 4.1: Zmienne lingwistyczne powstałe w wyniku analizy histogramów



Rysunek 4.4: Struktura sieci C-MLP2LN dla danych *Iris*

$\mathcal{R}_1$: petal width = s $\vee$ petal length = s $\rightarrow$ *setosa*
$\mathcal{R}_2$: petal width = m $\wedge$ petal length = m $\rightarrow$ *versicolor*
$\mathcal{R}_3$: petal width = l $\vee$ petal length = l $\rightarrow$ *virginica*

Tylko dwie cechy (petal width i petal length) są używane w klasyfikacji – pozostałe dwie zostały wyeliminowane przez sieć. Pierwsza reguła klasyfikuje poprawnie wszystkie wektory z klasy *setosa*. Cały zestaw reguł klasyfikuje poprawnie 147 przypadków (98%).

Setosa	(0,0,0	0,0,0	+,0,0	+,0,0)	$\theta = 1$
Versicolor	(0,0,0	0,0,0	0,+,0	0,+,0)	$\theta = 2$
Virginica	(0,0,0	0,0,0	0,0,+	0,0,+)	$\theta = 1$

Tabela 4.2: Wagi i progi sieci C-MLP2LN dla danych *Iris*

Metody dyskretyzacji mogą być także bardzo przydatne również wtedy, gdy stosujemy sieci z jednostkami L. Kiedy startujemy od przypadkowych parametrów jednostek L sieć wymaga długiego okresu uczenia. Znacznie szybciej można znaleźć dobre rozwiązanie, jeśli zainicjujemy parametry jednostek L na podstawie analizy histogramów czy innej metody dyskretyzacji. W takiej sytuacji proces uczenia sieci poprawi początkowe ustawienia tak, żeby uzyskać stabilne i proste reguły. W zależności od sposobu używania parametrów regularyzacyjnych w trakcie trenowania sieci możemy uzyskać różne zestawy reguł. Na przykład wymuszając jak najprostsze reguły otrzymujemy zestaw, który klasyfikuje zbiór z dokładnością 96%:

$\mathcal{R}_1$: petal length $\leqslant 2.56 \rightarrow$ *setosa*
$\mathcal{R}_2$: petal width $> 1.63 \rightarrow$ *virginica*
$\mathcal{R}_3$: *else* $\rightarrow$ *versicolor*

Stosując mniejsze wartości parametru odpowiedzialnego za zerowanie wag możemy dostać nieco bardziej złożone, ale i bardziej dokładne zestawy reguł.

Opisy złożoności i dokładności zestawów reguł stworzonych przez różne systemy zostały zebrane w tabeli 4.3.

Metoda	Liczba reguł/przesłanek/cech	Typ	Dokładność
ReFuNN [104]	9/26/4	F	95.7
ReFuNN [104]	14/28/4	F	95.7
ReFuNN [104]	104/368/4	F	95.7
Grobian [26]	118/?/4	R	100
GA+NN [95]	6/6/4	W	100
NEFCLASS[137]	7/28/4	F	96.7
NEFCLASS[137]	3/6/2	F	96.7
FuNe-I[82]	7/?/3	F	96.0
C-MLP2LN	2/2/1	C	95.7
C-MLP2LN	2/2/2	C	96.0
C-MLP2LN	2/3/2	C	98.0
SSV	2/3/2	C	98.0

Tabela 4.3: Reguły logiczne dla danych *Iris*. Typy reguł: F=Fuzzy (rozmyte), C=Crisp (ostre), R=Rough (przybliżone), W=Weighted (ważone)

Mushroom. Wzorcowym przypadkiem pokazującym, że reguły generowane są od najbardziej ogólnych do najbardziej szczegółowych jest przykład danych o

grzybach. Każdy wektor opisany jest 22 symbolicznymi atrybutami, przyjmującymi w sumie 125 wartości. Nie trzeba więc tutaj przeprowadzać dyskretyzacji ani konstruować sieci z jednostkami L. Wygenerowane reguły i liczby obejmowanych przypadków prezentuje tabela 4.4. Kolejność reguł odpowiada kolejności ich generowania przez sieć – warto zwrócić uwagę na wzrastający poziom szczegółowości.

odor = $\neg$(almond $\vee$ anise $\vee$ none)	8004
spore-print-color = green	72
odor=none $\wedge$ stalk-surface-below-ring=scaly $\wedge$ stalk-color-above-ring = $\neg$brown	40
habitat = leaves $\wedge$ cap-color = white	8

Tabela 4.4: Reguły dla danych o grzybach z liczbami obejmowanych przypadków

Ten zestaw reguł jest jednym z najprostszych opisów badanego zbioru danych i klasyfikuje poprawnie wszystkie przypadki ze zbioru treningowego. Choć zbiór z danymi o grzybach wydaje się być dość łatwym do klasyfikacji (także na losowo wybranej próbce złożonej z 10% wektorów uzyskaliśmy ten sam zestaw reguł), to jednak nie wszystkie systemy tworzące opisy regułowe tworzą tak zwarte i dokładne reguły. Zestawienie znanych nam wyników dla tych danych przedstawia tabela 4.5.

Hypothyroid. Dla zbioru danych *Hypothyroid* metoda MLP2LN znalazła zestaw 4 reguł, które klasyfikują poprawnie 99.68% wektorów treningowych i 99.07% testowych. Do opisania pierwszej klasy wystarczyły dwie reguły, drugą klasę opisuje jedna, natomiast trzecia klasa jest reprezentowana jako uzupełnienie sumy reguł dla pierwszych dwóch klas:

$\mathcal{R}_1$: FTI $< 63 \wedge$ TSH $\geqslant 29 \rightarrow$ *primary hypothyroid*

$\mathcal{R}_2$: FTI $< 63 \wedge$ TSH $\in [6.1, 29) \wedge$ T3 $< 20 \rightarrow$ *primary hypothyroid*

$\mathcal{R}_3$: FTI $\in [63, 180] \wedge$ TSH $\geqslant 6.1 \wedge$ on thyroxine=no $\wedge$ surgery=no $\rightarrow$ *compensated hypothyroid*

$\mathcal{R}_4$: *else normal*

Poprawności klasyfikacji różnych metod dla zbioru *Hypothyroid* zostały przedstawione w tabeli 3.2 na stronie 97.

Metoda	Liczba reguł/przesłanek/cech	Dokładność
RULENEG[162]	300/8087	91.0
REAL [36]	155/6603	98.0
DEDEC [176]	26/26	99.8
TREX[3]	3/13	100
C4.5 (decision tree)	3/3	99.8
RULEX[4]	1/3/1	98.5
Successive Regulariz.[46]	1/4/2	99.4
Successive Regulariz.[46]	2/22/4	99.9
Successive Regulariz.[46]	3/24/6	100
C-MLP2LN, SSV	1/3/1	98.5
C-MLP2LN, SSV	2/4/2	99.4
C-MLP2LN	3/7/4	99.9
SSV	3/7/4	99.9
C-MLP2LN	4/9/6	100
SSV	4/9/5	100

Tabela 4.5: Zestawienie wyników systemów regułowych dla danych *Mushroom*

4.2 Szukanie optymalnego MLP

Metody minimalizacji i metody szukania mają wspólny cel polegający na znalezieniu minimalnej wartości funkcji kosztów. Dlatego też można zastąpić metody gradientowe odpowiednimi metodami szukania. W praktyce, aby takie przeszukiwanie trwało sensownie krótko, musi ono ograniczać się do stosunkowo małego podzbioru przestrzeni wszystkich możliwych rozwiązań. W przypadku szukania parametrów sieci MLP (zwłaszcza, kiedy jesteśmy zainteresowani uzyskaniem sieci, którą będzie można zinterpretować logicznie) można znacznie przyspieszyć szukanie ograniczając wartości wag do zbioru liczb całkowitych. W takim przypadku już proste wyszukiwanie algorytmem „najpierw najlepszy” jest często w stanie szybko znaleźć bardzo dobre rozwiązanie.

Sieci S-MLP (ang. *Search-based MLP*) wymagają by dane treningowe były zdefiniowane wyłącznie cechami symbolicznymi. Jeśli przestrzeń klasyfikacji ma ciągłe cechy, to trzeba je poddać dyskretyzacji, co tak jak w wielu innych systemach mocno wpływa na kształt ostatecznych wyników oraz zmniejsza wszechstronność algorytmu. Każdy z wektorów kodowany jest (tak jak w metodzie MLP2LN opisanej w rozdziale 4.1) wektorem o współrzędnych równych 1, bądź -1 (ewentualnie 0 dla wskazania, że wartość nie jest znana). Liczba wejść sieci jest zatem sumą liczb możliwych wartości symbolicznych dla wszystkich

cech. Wyjścia sieci wzajemnie jednoznacznie odpowiadają klasom, a każdy neuron warstwy ukrytej (realizującej główny cel generowania reguł) jest dedykowany konkretnej klasie, a więc jest połączony tylko z jednym wyjściem. Można więc taką sieć traktować jak kilka osobnych sieci (po jednej dla każdej z klas), zwłaszcza, że procesy uczenia dla poszczególnych wyjść są całkowicie niezależne. Najskuteczniesze jest podejście konstruktywistyczne, w którym zaczynamy od uczenia pojedynczego neuronu i w miarę potrzeb dodajemy następne, choć można również rozważać architektury statyczne, w których mamy ustaloną liczbę wektorów ukrytych dla każdego z wejść – jeśli dodamy więcej neuronów niż jest to konieczne dla stworzenia odpowiedniego opisu regułowego, to algorytm uczenia wyeliminuje niepotrzebne neurony.

Algorytm S-MLP [51] rozpoczyna działanie od zainicjowania wartości wszystkich wag $W_{ij} = 0$ i progów $\theta_i = -0.5$. Takie ustawienia powodują, że neuron wyjściowy nie wzbudzi się dla żadnej kombinacji wejść. Następnie ustala się wartość kroku Δ, o który będą się zmieniały wagi i progi. W każdej iteracji procesu szukania rozpatruje się wszystkie możliwe zmiany wag i progów ($W_{ij} \pm \Delta,\ \theta_i \pm \Delta$) i ocenia wpływ tych zmian na błąd klasyfikacji. Można tu zastosować wiele różnych algorytmów np. „najpierw najlepszy" albo „przeszukiwanie wiązką" [102] dla pojedynczych zmian wag w każdym kroku. Nie zawsze jednak tak proste metody szukania są w stanie znaleźć satysfakcjonujące minimum lokalne funkcji błędu. Dlatego też można zastosować bardziej złożone obliczeniowo metody (np. zmieniając w pojedynczym kroku dowolną parę parametrów). Aby przyspieszyć działanie metody i jednocześnie uzyskiwać dobre wyniki stosowaliśmy szukanie dwuetapowe: w pierwszym etapie zmienialiśmy wagi i progi pojedynczo, by wybrać te z nich, które dają najlepszą zmianę błędu klasyfikacji, a w drugim rozpatrywaliśmy wszystkie możliwe pary (a nawet podzbiory) złożone z parametrów wybranych w pierwszym etapie.

W tej metodzie można też zastosować dodatkową technikę ułatwiającą dochodzenie do optymalnych rozwiązań przez stopniowe zwiększanie rozdzielczości, w której „oglądamy" przestrzeń (tzn. startujemy ze stosunkowo dużą wartością Δ i w trakcie procesu szukania stopniowo ją zmniejszamy). Taki sposób szukania optymalnej sieci można porównać do stopniowego zmniejszania parametru uczenia w metodzie propagacji wstecznej błędu, czy też do technik „stopniowego schładzania".

Nakładając na metodę dodatkowe ograniczenia możemy tworzyć sieci, które z łatwością będzie można opisać ostrymi albo rozmytymi regułami logicznymi. Jeśli na przykład wszystkie wagi w sieci będą liczbami całkowitymi (co otrzymamy stosując $\Delta = 1$) i funkcje realizowane przez neurony ukryte są wystarczająco stromymi sigmoidami, to stworzona sieć może być opisana przez zbiór reguł (wyniki prezentowane niżej mają formę reguł logiki klasycznej, choć prostsze opisy można uzyskać z wykorzystaniem reguł typu *M*-z-*N*). Reguły tworzone są w wy-

niku analizy wszystkich możliwych kombinacji sygnałów wejściowych (tak samo jak w przypadku metody MLP2LN). Aby zagwarantować sobie małą liczbę reguł logicznych można dodatkowo ograniczyć przeszukiwaną przestrzeń przez wyłączenie ze zmian wartości progów i automatyczne ustawianie każdego z nich po każdej zmianie wag tak, by był równy sumie wszystkich wag połączeń dochodzących do jego neuronu pomniejszonej o 0.5 ($\theta_i = \sum_j W_{ij} - 0.5$). W takim przypadku każdy z neuronów ukrytych będzie mógł być opisany pojedynczą regułą logiki klasycznej jako, że tylko jedna kombinacja wejść da w sumie wartość przewyższającą wartość progu.

4.2.1 Rezultaty

Metoda S-MLP została zastosowana do kilku testowych zbiorów danych, a uzyskane wyniki potwierdziły, że całkiem proste metody szukania w ograniczonej przestrzeni sieci MLP, mimo, że po nałożeniu ograniczeń tracą własności uniwersalnych aproksymatorów, to są w stanie uzyskać bardzo atrakcyjne rezultaty.

Testy klasyfikacji. S-MLP bez ograniczeń mających na celu łatwe generowanie prostych reguł logicznych, jest także bardzo skutecznym systemem klasyfikacji.

10-krotny test kroswalidacyjny dla zbioru *Iris* daje 96% poprawności. Jest to wynik w pełni zadowalający, ponieważ wiadomo, że w tym zbiorze jest kilka wektorów należących do klas *versicolor* oraz *virginica*, które są bardzo podobne i ich rozróżnianie jest przejawem nadmiernego dopasowania do danych.

Dla danych *Ljubljana breast cancer* metoda S-MLP znalazła rozwiązanie dające 47 błędów (dokładność 83.6%). W związku z informacjami na temat tego zbioru danych uzyskanymi innymi metodami (p. rozdział 6), trudno oczekiwać po tym rozwiązaniu dobrej generalizacji.

W przypadku zbioru *Appendicitis* S-MLP daje rezultaty podobne do najlepiej spisujących się na tych danych systemów. W teście *leave-one-out*, uzyskujemy poprawność 89.7%, która jest wyższa od wyników zaprezentowanych przez Weiss'a [182]. Wynik ten nie został jednak uwzględniony w tabeli 3.1 umieszczonej na stronie 95, ponieważ dotyczy on zdyskretyzowanych danych, więc jego porównywanie z innymi może nie być sprawiedliwe (komentarze na ten temat znajdują się w rozdziale 3.6).

Indukcja reguł. Wersja S-MLP specjalizująca się w indukcji reguł logicznych pozwoliła znaleźć interesujące zestawy reguł dla użytych w testach danych.

Opis danych *Iris* to trzy proste reguły dające zaledwie 4 błędy, czyli 97.3% poprawności reklasyfikacji:

$\mathcal{R}_1$: petal width $\leqslant 0.9 \rightarrow$ *setosa*

$\mathcal{R}_2$: petal length > 4.9 $\rightarrow$ *virginica*
$\mathcal{R}_3$: petal width > 1.7 $\rightarrow$ *virginica*
$\mathcal{R}_4$: *else versicolor*

Jest to zestaw bardzo podobny do tych otrzymanych metodami SSV i MLP2LN – wszystkie są bardzo trafnymi opisami.

Dla danych *Ljubljana breast cancer* zastosowana została dwuetapowa wersja algorytmu S-MLP. W pierwszym etapie wyznaczanych było osiem najatrakcyjniejszych wag, by potem przetestować ich wszystkie możliwe kombinacje. W ten sposób znaleźliśmy dwie reguły logiczne dla klasy *recurrence-events*:

$\mathcal{R}_1$: inv-nodes $\neq$ 0-2 $\wedge$ deg-malig = 3 $\wedge$ tumor-size $\neq$ 45-49 $\rightarrow$ *recurrence-events*
$\mathcal{R}_2$: inv-nodes = [9,11] $\wedge$ tumor-size = [35-39] $\rightarrow$ *recurrence-events*
$\mathcal{R}_3$: *else no-recurrence-events*

Poprawność tych reguł to 77.6% (64 błędy).

Uproszczona wersja tych reguł:

$\mathcal{R}_1$: inv-nodes $\neq$ 0-2 $\wedge$ deg-malig = 3 $\rightarrow$ *recurrence-events*
$\mathcal{R}_2$: *else no-recurrence-events*

daje dokładność 77%. Jest to ta sama reguła, która została znaleziona przy użyciu drzewa SSV. Różnica w ocenie poprawności (p. rozdział 6) wynika z innego traktowania wartości brakujących – dla sieci S-MLP braki były traktowane jak dodatkowa wartość cechy.

Dokładniejsza metoda szukania (zmiana dwóch wag jednocześnie) pozwoliła znaleźć inny, bardziej dokładny zestaw reguł (78.3% – 62 błędy):

$\mathcal{R}_1$: inv-nodes $\neq$ 0-2 $\wedge$ deg-malig = 3 $\wedge$ tumor-size $\neq$ 45-49 $\wedge$ age $\neq$ 10-19 $\rightarrow$ *recurrence-events*
$\mathcal{R}_2$: inv-nodes = [9,11] $\wedge$ age $\neq$ 40-49 $\rightarrow$ *recurrence-events*
$\mathcal{R}_3$: tumor-size = [35-39] $\wedge$ age = 30-39 $\rightarrow$ *recurrence-events*
$\mathcal{R}_4$: *else no-recurrence-events*

Dokładność i złożoność reguł zależy od tego którą klasę opisujemy regułami, a którą pozostawiamy jako warunek domyślny. Dla klasy „no-recurrence-events” S-MLP znajduje jeszcze dokładniejsze reguły (59 błędów, 79.4% poprawności), jednak zarówno ten jak i przedstawiony powyżej zestaw reguł są nadmiernie dopasowane do zbioru treningowego.

Zbiór *Appendicitis* został opisany przez metodę S-MLP regułą o poprawności 92.5%. Dodanie drugiej reguły produkuje następujący zestaw:

$\mathcal{R}_1$: MBAP $\notin$ (12.6,18.9] $\vee$ MBAP > 25.2) $\wedge$ MNEA $\leqslant$ 6650 $\wedge$ WBC1 $\notin$ (16775,17400] $\rightarrow$ *class 1*

$\mathcal{R}_2$: MNEP $\notin$ (71.6,82.4] $\wedge$ MBAP $\in$ (1,3]$\wedge$ WBC1 > 9525 $\rightarrow$ *class 1*

$\mathcal{R}_3$: *else class 2*

Jego dokładność to 94.3%, więc druga reguła poprawia wynik dzięki poprawnej klasyfikacji dwóch dodatkowych wektorów, co wobec jej trzech przesłanek jest wyrazem nadmiernej szczegółowości.

Rozdział 5

Dodatkowe algorytmy

5.1 Optymalizacja reguł

Ponieważ metody gradientowe odnajdują zwykle lokalne minima funkcji błędu, zbiór reguł „wyczytany” ze struktury sieci neuronowej nie musi być optymalnym opisem danych treningowych. Jednak mając tak stworzony opis regułowy można próbować go udoskonalać modyfikując granice przedziałów pojawiających się w regułach przy użyciu globalnych metod optymalizacji. Takie modyfikacje można przeprowadzać na wiele różnych sposobów [43], np. można maksymalizować ślad macierzy rozrzutu $P(C_i,C_j|M)$ aby uzyskać maksymalną poprawność klasyfikacji. Można też minimalizować liczbę pomyłek klasyfikacji kosztem częściej udzielanych przez reguły odpowiedzi „nie wiem”, aby w ten sposób zwiększać wiarygodność klasyfikacji dla tych przypadków, które spełniają przesłanki otrzymanych reguł. Dobre wyniki daje kombinacja tych dwóch metod, czyli optymalizacja funkcji błędu dla klasyfikatora regułowego:

$$E(M) = \gamma \sum_{i \neq j} P(C_i,C_j|M) - \mathrm{Tr}\, P(C_i,C_j|M) \geqslant -n \tag{5.1}$$

gdzie n to liczba klasyfikowanych wektorów, M to parametry modelu (dla reguł przedziały zmiennych lingwistycznych), zaś γ określa balans pomiędzy poziomem zaufania do reguł a liczbą wektorów odrzucanych jako nieznane. Można też stworzyć hierarchiczny system reguł tak, aby przy użyciu pewnych zestawów reguł dawać bardzo wiarygodne (znamienne) odpowiedzi, a przy użyciu innych mniej wiarygodne, ale bardziej wrażliwe, czyli obejmujące coraz większe części przestrzeni danych [43, 45].

Reguły logiki klasycznej traktowane jako funkcje przynależności są funkcjami dwuwartościowymi, więc z braku ciągłości minimalizacja błędu 5.1 metodami gradientowymi nie wchodzi w rachubę. Z pomocą przychodzą wówczas metody szukania. Stosunkowo szybkim i zarazem dokładnym algorytmem, który świet-

nie nadaje się do tego celu jest metoda adaptacyjnego symulowanego wyżarzania (ASA – *Adaptive Simulated Annealing* – p. rozdział 2.4.2).

5.1.1 Rezultaty

Metoda ASA minimalizując wartość błędu określonego wzorem 5.1 dla reguł uzyskanych metodą *C-MLP2LN* dla zbioru *Hypothyroid* (p. rozdział 4.1.1) znalazła następujący (nieco dokładniejszy) zestaw reguł:

$\mathcal{R}_1$: TSH $\geqslant 30.48 \wedge$ FTI $< 64.27 \rightarrow$ *primary hypothyroid*
$\mathcal{R}_2$: TSH $\in [6.02, 29.53] \wedge$ FTI $< 64.27 \wedge$ T3 $< 23.22 \rightarrow$ *primary hypothyroid*
$\mathcal{R}_3$: TSH $\geqslant 6.02 \wedge$ FTI $\in [64.27, 186.71] \wedge$ TT4 $\in [50, 150.5) \wedge$ on thyroxine=no $\wedge$ surgery=no $\rightarrow$ *compensated hypothyroid*
$\mathcal{R}_4$: *else normal*

Tak poprawione reguły klasyfikują błędnie tylko 4 wektory ze zbioru treningowego (99.89% poprawności) i 22 ze zbioru testowego (99.36% poprawności). Bardzo podobny zestaw znaleźli Weiss i Kapouleas używając heurystycznej wersji metody PVM [182]. Podobnej jakości i złożoności są również reguły uzyskane poprzez drzewa SSV (p. rozdział 6).

5.2 Rozmywanie reguł logiki klasycznej

Stosowanie hierarchicznych systemów reguł daje pewne możliwości oceny prawdopodobieństwa poprawności klasyfikacji. Podobnie jak w przypadku większości metod określających prawdopodobieństwa przynależności do klas na podstawie drzew decyzji (p. rozdział 3.10), otrzymamy tu prawdopodobieństwa „makrolokalne" tzn. dla pewnego obszaru a nie dla zadanego punktu w przestrzeni.

Aby stosując klasyfikację regułową mieć możliwość przypisania podejmowanej decyzji współczynnika zaufania, można zamienić klasyczne reguły na rozmyte przez „złagodzenie" granic decyzji. Alternatywnym sposobem jest uznanie niedokładności danych i uwzględnienie stosownych rozkładów przy obliczaniu prawdopodobieństwa spełniania poszczególnych reguł klasyfikacji przez dany przypadek i do poszczególnych klas. Zastępując wartość x danej cechy gaussowskim rozkładem $G(y; x, s_x)$, można obliczyć prawdopodobieństwo przynależności wartości tej cechy do przedziału występującego w regule:

$$P(x \in (a,b)) = \frac{1}{2}\left[\mathrm{erf}\left(\frac{b-x}{s_x\sqrt{2}}\right) - \mathrm{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right]. \tag{5.2}$$

Jest ono dane przez funkcję błędu erf (dystrybuantę standardowego rozkładu normalnego), która bardzo przypomina funkcje logistyczne, używane w sieciach neuronowych. Przyjmując założenie niezależności rozkładów cech wykorzystywanych w regule można wyliczyć prawdopodobieństwo spełniania reguły przez dany wektor jako iloczyn prawdopodobieństw przynależności wartości odpowiednich cech do przedziałów zadanych regułą. W efekcie połączenie rozmytych gaussowsko danych z regułami logiki klasycznej daje ten sam efekt, co użycie ostro określonych danych z rozmytymi regułami, których funkcje przynależności mają kształt okienek zdefiniowanych jako różnica albo iloczyn stosownych sigmoid. Przyjęcie założenia niezależności cech jest pewną naiwnością, jednak w porównaniu z techniką Naiwnego Klasyfikatora Bayesowskiego jest ono tutaj bardziej uzasadnione, bowiem opiera się na założeniu, że reguły klasyfikacji zostały wygenerowane z dbałością o jak najbardziej zwartą formę, co w pewnym stopniu zabezpiecza przed używaniem w tej samej regule przesłanek mocno od siebie zależnych.

Uznanie danych za rozmyte i stosowanie prawdopodobieństw zamiast binarnych decyzji może samo w sobie przynieść poprawę klasyfikacji, daje więcej informacji w przypadku trudnych do sklasyfikowania przypadków, a także daje możliwość zastosowania metod gradientowych do optymalizacji zbiorów reguł [45]. Parametry rozmyć s_x można dla niektórych danych ocenić na podstawie dokładności pomiarów. Można również uznać je za parametry adaptacyjne procesu minimalizacji funkcji 5.1.

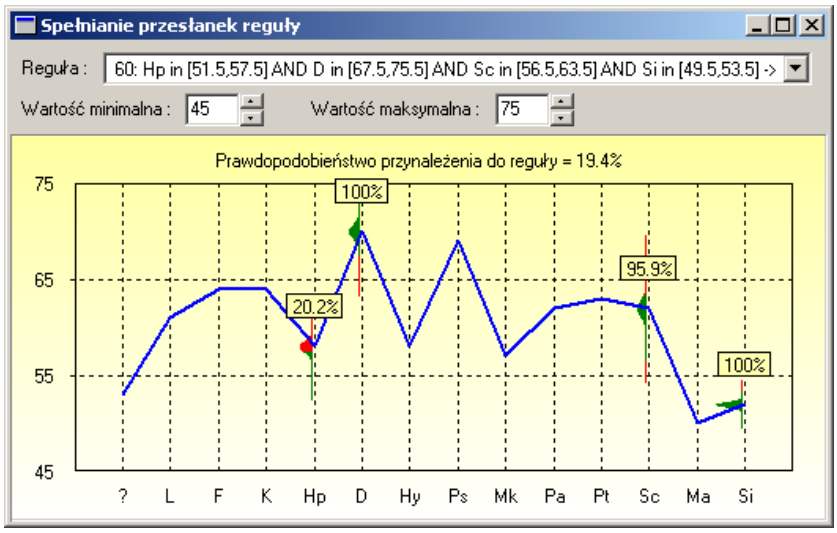
5.2.1 Rezultaty

Technika rozmywania reguł została zastosowana w systemie wspomagającym podejmowanie decyzji w zakresie interpretacji wyników testów psychometrycznych. W szczególności badaniom zostały poddane wyniki testu MMPI (ang. *Minnesota Multiphasic Personality Inventory*), który jest jednym z najbardziej uznanych i najczęściej stosowanych przez psychometrów testów. Jest to test służący klasyfikacji psychiatrycznych typów nozologicznych.

Test MMPI polega na zebraniu odpowiedzi na ok. 550 pytań i wyliczeniu na ich podstawie wartości czterech skal kontrolnych oraz dziesięciu klinicznych.

W przychodni psychologicznej Uniwersytetu Mikołaja Kopernika zebrano wyniki testów dla ponad tysiąca kobiet i podobnej liczby mężczyzn. Dane te zostały przeanalizowane przez psychologów i sklasyfikowane do 27 klas w przypadku kobiet oraz 28 w przypadku mężczyzn.

Na podstawie tych wyników stworzone zostały (systemem FSM [50, 47]) reguły klasyfikacji, które stały się bazą systemu wspomagania decyzji. Jednym z głównych sposobów interpretacji zaimplementowanych w tym systemie było właśnie rozmywanie reguł (a właściwie danych). Przykład analizy przypadku w kon-



Rysunek 5.1: Analiza prawdopodobieństwa spełniania reguły przy założeniu niepewności danych

tekście jednej z reguł przedstawiony jest na rysunku 5.1. Rysunek ten pokazuje profil danego pacjenta (jako linię łączącą wartości kolejnych skal) oraz jedną z reguł klasyfikacji. Dla każdej cechy użytej w regule wykreślona jest krzywa obrazująca rozkład danej wartości – część oznaczona kolorem zielonym obrazuje zakres stosownej przesłanki reguły, a kolorem czerwonym zakres wartości nie pasujących do reguły. Określone są prawdopodobieństwa przynależności wartości skal do przesłanek występujących w regule, oraz prawdopodobieństwo spełnienia reguły przez przypadek wyliczone jako iloczyn prawdopodobieństw dla poszczególnych skal.

Taki sposób interpretacji pozwala uwzględnić nie tylko te reguły, w które przypadek wpada w sensie logiki klasycznej, ale także te, których prawdopodobieństwo spełnienia jest niezerowe jeśli założy się odpowiednie rozmycie wyliczonych z testu skal. Przypadek przedstawiony na rysunku obrazuje regułę, która w klasycznej analizie zostałaby dla ilustrowanego przypadku zignorowana, lecz w praktyce jest ona warta uwagi, bo tylko jedna z czterech przesłanek nie jest w satysfakcjonującym stopniu spełniona.

Rozdział 6

Podsumowanie

W rozprawie podjęto szereg aspektów problemów klasyfikacji obiektów oraz odkrywania wiedzy na podstawie baz danych. Przedstawiono wyniki badań, które miały na celu opracowanie efektywnych obliczeniowo a zarazem dokładnych metod budowania klasyfikatorów, które można przedstawić w czytelnej postaci. Skupiono się na metodach indukcji reguł korzystających bezpośrednio z danych (w przeciwieństwie do prób opisywania regułami gotowych klasyfikatorów), jako dających większe możliwości odkrycia dokładnego opisu klas obiektów. Jako język opisu klasyfikacji przyjęto logikę klasyczną pierwszego rzędu, której formuły są najłatwiejsze do zrozumienia i interpretacji.

Przedstawiono opracowane kryterium separowalności SSV, które znalazło skuteczne zastosowania do budowania binarnych drzew decyzji, a za ich pośrednictwem do indukcji reguł klasyfikacji, a także do zadań transformacji danych, bardzo użytecznych w budowaniu hybrydowych systemów klasyfikacji.

Zaprezentowano także algorytmy generowania reguł klasyfikacji wykorzystujące struktury oraz algorytmy uczenia sieci neuronowych.

Przedstawione metody stanowią łącznie bogaty system analizy danych pozwalający na rozwiązywanie zadań klasyfikacji, przedstawianie zrozumiałych opisów tworzonych modeli, a także wspomaganie klasyfikacji nowych przypadków z możliwością wyjaśniania diagnoz regułami logiki klasycznej oraz prawdopodobieństwami ich spełniania.

Badania wykonane przez autora. Definicja kryterium separowalności SSV, oraz wszystkie, zaprezentowane w rozprawie, algorytmy, które je wykorzystują są efektem badań zrealizowanych przez autora, który samodzielnie zaimplementował wszystkie te metody (indukcji drzew decyzji i reguł klasyfikacji, generowania drzew heterogenicznych i lasów, dyskretyzacji oraz uciąglania danych, selekcji cech, a także rozmywania reguł/danych). Autor poddał stworzone algorytmy analizie i testom porównawczym (wyjątki stanowią testy metody selekcji cech,

które były wykonane wspólnie z Włodzisławem Duchem, Jackiem Biesiadą oraz Tomaszem Winiarskim oraz testy metody uciąglania danych, wykonane wspólnie z Norbertem Jankowskim).

Prace nad metodami indukcji reguł wykorzystującymi struktury i algorytmy sieci neuronowych autor wykonywał wspólnie z Włodzisławem Duchem i Rafałem Adamczakiem.

Metody oparte na kryterium SSV. Drzewa decyzji budowane na bazie kryterium separowalności SSV, którego definicja jest bliska intuicyjnemu (potocznemu) rozumieniu separowalności, dowiodły w testach, że mogą klasyfikować dane równie dokładnie (a często dokładniej) niż inne systemy klasyfikacji. Dla wielu zadań odkryły bardzo zwięzłe i dokładne opisy danych, tak pożądane w wielu dziedzinach życia.

Algorytmy tworzenia lasów oraz heterogenicznych drzew pozwalają opisać jeden zbiór danych wieloma różnorodnymi, alternatywnymi zestawami czytelnych reguł, które mogą być bardzo pomocne w próbach „zrozumienia danych".

W oparciu o kryterium SSV opracowano również metody dyskretyzacji oraz uciąglania danych, które dzięki dużej szybkości działania dają się (bez nadmiernych kosztów) stosować w połączeniu z innymi systemami klasyfikacji, które potrzebują specyficznej formy danych. Systemy złożone, których części stanowiły te metody, podobnie jak i metoda selekcji cech przy pomocy drzew decyzji, zostały z bardzo dobrym skutkiem zastosowane do realnych problemów klasyfikacji.

Metody sieciowe. Również metody oparte na sieciach neuronowych (MLP2LN oraz S-MLP) dowiodły, że w wielu problemach można znaleźć reguły klasyfikacji działające równie dokładnie jak najlepsze klasyfikatory nieregułowe. Co więcej, dla niektórych danych medycznych reguły logiczne odkryte za pomocą sieci neuronowych są znacznie dokładniejsze niż same sieci oraz wiele innych klasyfikatorów.

Końcowe wnioski. Przedstawione metody indukcji drzew decyzji SSV i reguł logicznych na podstawie baz danych pozwalają w sposób efektywny generować dokładne, a zarazem zwarte i zrozumiałe opisy. Pozwoliły odkryć interesujące opisy logiczne dla licznych problemów klasyfikacji. W wielu zadaniach uzyskane reguły logiczne są równie dokładne jak systemy działające jako *czarne skrzynki* tzn. nie udostępniające czytelnego opisu mechanizmu klasyfikacji. W niektórych problemach są nawet dokładniejsze od innych metod, wskazując na wyraźną logiczną strukturę klas. Może ona odzwierciedlać sposób podejmowania decyzji przez ekspertów. Ujawnianie logicznej postaci zasad według których diagnozują eksperci jest jednym z głównych celów dziedziny zajmującej się wydobywaniem

wiedzy z danych. Jest nie tylko spektakularne, ale przede wszystkim bardzo wartościowe – zwłaszcza w przypadkach, kiedy sami eksperci nie potrafią w prosty sposób zwerbalizować swoich strategii podejmowania decyzji.

Również pozostałe narzędzia oparte na kryterium SSV mogą być bardzo pomocne w odkrywaniu wiedzy na temat danych, bo np. umożliwiają stosowanie metod wymagających przestrzeni klasyfikacji o cechach wyłącznie symbolicznych bądź wyłącznie ciągłych.

Liczne bardzo atrakcyjne rezultaty nie oznaczają jednak, że systemy regułowe są w stanie zawsze skuteczniej klasyfikować dane niż inne metody. Żaden algorytm nie jest lepszy od wszystkich innych we wszelkich zastosowaniach, więc w konkretnym przypadku należy zwykle sięgnąć po kilka metod i porównać ich wyniki. Różne problemy klasyfikacji wymagają różnych granic decyzji, więc i zastosowania różnych systemów klasyfikacji. W sytuacjach, w których reguły logiki klasycznej dają wyniki statystycznie tożsame z najlepszymi osiąganymi dla danego zadania, opisy logiczne mogą być zupełnie wystarczające – w innych przypadkach warto sięgnąć również po inne metody by odkryć naturę problemu.

Dodatek A

Lista publikacji i opracowań autora

Prace dotyczące kryterium SSV

Publikacje w materiałach konferencyjnych (recenzowane):

1. K. Grąbczewski, W. Duch. Heterogeneous forests of decision trees. *Proceedings of International Conference on Artificial Neural Networks*, wolumen 2415 serii *Lecture Notes in Computer Science*, strony 504–509. Springer, 2002.

2. K. Grąbczewski, W. Duch. Forests of decision trees. *Proceedings of International Conference on Neural Networks and Soft Computing*, Advances in Soft Computing, strony 602–607. Physica-Verlag (Springer), 2002.

3. W. Duch, J. Biesiada, T. Winiarski, K. Grudziński, K. Grąbczewski. Feature selection based on information theory filters and feature elimination wrapper methods. *Proceedings of the International Conference on Neural Networks and Soft Computing (ICNNSC 2002)*, Advances in Soft Computing, strony 173–176, Zakopane, 2002. Physica-Verlag (Springer).

4. W Duch, K. Grąbczewski. Heterogeneous adaptive systems. *Proceedings of the World Congress of Computational Intelligence*, Honolulu, Maj 2002.

5. W. Duch, N. Jankowski, K. Grąbczewski, R. Adamczak. Optimization and interpretation of rule-based classifiers. *Proceedings of IIS 2000*, Advances in Soft Computing, strony 1–13. Physica-Verlag (Springer), 2000.

6. W. Duch, K. Grąbczewski, R. Adamczak, K. Grudziński, Z. S. Hippe. Rules for melanoma skin cancer diagnosis. *KOSYR 2001*, strony 59–68, Wrocław, 2001.

7. K. Grąbczewski, W. Duch. A general purpose separability criterion for classification systems. *Proceedings of the 4th Conference on Neural Networks and Their Applications*, strony 203–208, Zakopane, Poland, Czerwiec 1999.

8. K. Grąbczewski, W. Duch. The separability of split value criterion. *Proceedings of the 5th Conference on Neural Networks and Their Applications*, strony 201–208, Zakopane, Poland, Czerwiec 2000.

Prace zgłoszone na konferencje:

1. K. Grąbczewski, N. Jankowski. Symbolic data transformations for continuous data oriented models. Praca zgłoszona na konferencję ICANN 2003.

Prace nt. indukcji reguł z wykorzystaniem sieci neuronowych

Publikacje w wydawnictwach naukowych:

1. W. Duch, R. Adamczak, K. Grąbczewski. Methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.

2. K. Grąbczewski, W. Duch, R. Adamczak. Neuronowe metody odkrywania wiedzy w danych. W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, redaktorzy, *Biocybernetyka 2000, Tom 6: Sieci neuronowe*, rozdział III.20, strony 637–662. Akademicka Oficyna Wydawnicza EXIT, 2000.

3. W. Duch, R. Adamczak, K. Grąbczewski, N. Jankowski. Neural methods of knowledge extraction. *Control and Cybernetics*, 29(4):997–1018, 2000.

4. W. Duch, R. Adamczak, K. Grąbczewski, G. Żal, Y. Hayashi. Fuzzy and crisp logical rule extraction methods in application to medical data. P. S. Szczepaniak, P. J. G. Lisboa, J. Kacprzyk, redaktorzy, *Fuzzy Systems in Medicine*, strony 593–616. Physica-Verlag, Springer, 2000.

5. W. Duch, R. Adamczak, K. Grąbczewski, G. Żal. Hybrid neural-global minimization method of logical rule extraction. *Int. Journal of Advanced Computational Intelligence*, 1999.

6. W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:211–219, 1998.

Publikacje w materiałach konferencyjnych (recenzowane):

1. W. Duch, R. Adamczak, K. Grąbczewski. Optimization of logical rules derived by neural procedures. *Proceedings of the International Joint Conference on Neural Networks*, Washington, Lipiec 1999. paper no. 741.

2. W. Duch, R. Adamczak, K. Grąbczewski. Neural optimization of linguistic variables and membership functions. *Proceedings of ICONIP'99*, Perth, Australia, 1999.

3. W. Duch, R. Adamczak, K. Grąbczewski. Neural methods for analysis of psychometric data. *Proceedings of the International Conference EANN'99*, strony 45–50, Warsaw, 1999.

4. W. Duch, R. Adamczak, K. Grąbczewski, G. Żal. A hybrid method for extraction of logical rules from data. *Proceedings of the Second Polish Conference on Theory and Applications of Artificial Intelligence*, strony 61–82, Łódź, 1998.

5. W. Duch, R. Adamczak, K. Grąbczewski, G. Żal. Hybrid neural-global minimization logical rule extraction method for medical diagnosis support. *Proceedings of IIS'98*, strony 85–94, Malbork, Poland, 1998.

6. W. Duch, R. Adamczak, K. Grąbczewski, N. Jankowski, G. Żal. Medical diagnosis support using neural and machine learning methods. *Proceedings of the International Conference EANN'98*, strony 292–295, Gibraltar, 1998.

7. W. Duch, R. Adamczak, K. Grąbczewski, M. Ishikawa, H. Ueda. Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches. *Proc. of the European Symposium on Artificial Neural Networks (ESANN'97)*, strony 109–114, Bruge, 1997.

8. W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from medical datasets. *Proceedings of the Third Conference on Neural Networks and Their Applications*, strony 707–712, Kule, Październik 1997.

9. W. Duch, R. Adamczak, K. Grąbczewski. Logical rules for classification of medical data using ontogenic neural algorithm. solving engineering problems with neural networks. *Proceedings of the International Conference EANN'97*, strony 199–202, Stockholm, 1997.

10. W. Duch, R. Adamczak, K. Grąbczewski. Extraction of crisp logical rules using constrained backpropagation networks. *Proceedings of the International Conference on Artificial Neural Networks (ICNN'97)*, strony 2384–2389, Houston, 1997.

11. W. Duch, R. Adamczak, K. Grąbczewski. Constrained MLP and density estimation for extraction of crisp logical rules from data. *Proceedings of the ICONIP'97*, strony 831–834, New Zealand, Listopad 1997.

12. W. Duch, R. Adamczak, K. Grąbczewski. Constrained backpropagation for feature selection and extraction of logical rules. *Proceedings of the First Polish Conference on Theory and Applications of Artificial Intelligence (CAI'96)*, strony 163–170, Łódź, 1996.

13. W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from training data using backpropagation networks. *Proceedings of the 1st Online Workshop on Soft Computing*, strony 25–30, 1996.

14. W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from training data using backpropagation networks. *Proceedings of the First Polish Conference on Theory and Applications of Artificial Intelligence (CAI'96)*, strony 171–178, Łódź, 1996.

Inne prace

Publikacje w wydawnictwach naukowych:

1. L. Paulson, K. Grąbczewski. Mechanizing set theory: Cardinal arithmetic and the axiom of choice. *Journal of Automated Reasoning*, 17:291–323, 1996.

Publikacje w materiałach konferencyjnych (recenzowane):

1. W. Duch, K. Grąbczewski. Searching for optimal MLP. *Proceedings of the Fourth Conference on Neural Networks and Their Applications*, strony 65–70, Zakopane, Maj 1999.

Tutoriale:

1. W. Duch, R. Adamczak, K. Grąbczewski, Grudziński K, N. Jankowski, Naud N. Extraction of knowledge from data using computational intelligence methods. Tutorial – ICONIP 2000, 7th International Conference on Neural Information Processing, Dae-jong, Korea, Listopad 2000. (separate brochure, 54 pp).

2. W. Duch, R. Adamczak, K. Grąbczewski, Grudziński K, N. Jankowski, Naud N. Understanding the data: extraction, optimization and interpretation of logical rules. Tutorial – IJCNN 2000, International Joint Conference on Neural Networks, Listopad 2000. (separate brochure, 70 pp).

3. W. Duch, R. Adamczak, K. Grąbczewski, Grudziński K, N. Jankowski, Naud N. Extraction of knowledge from data using computational intelligence methods. Tutorial – International Conference on Artificial Neural Networks (ICANN), Vienna, Listopad 2001. (separate brochure, 63 pp).

Raporty techniczne:

1. L. Paulson, K. Grąbczewski. Mechanising set theory: Cardinal arithmetic and the axiom of choice. Raport instytutowy 377, Computer Laboratory, University of Cambridge, Sierpień 1996.

Prace zgłoszone na konferencje:

1. N. Jankowski, K. Grąbczewski. Toward optimal SVM. Praca zgłoszona na konferencję ICANN 2003.

Bibliografia

[1] R. Adamczak, W. Duch, N. Jankowski. New developments in the feature space mapping model. *Third Conference on Neural Networks and Their Applications*, strony 65–70, Kule, Poland, Październik 1997.

[2] J. A. Alexander, M. C. Mozer. Template-based algorithms for connectionist rule extraction. *Advances in Neural Information Processing Systems (NIPS 1994)*, 7:609–616, 1995.

[3] R. Andrews, J. Diederich, A. B. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8:373–389, 1995.

[4] R. Andrews, S. Geva. Rule extraction from a constrained error back propagation MLP. *Proc. 5th Australian Conference on Neural Networks, Brisbane*, strony 9–12, Queensland, 1994.

[5] R. Andrews, S. Geva. Rules and local function networks. R. Andrews, J. Diederich, redaktorzy, *Proc. of the Rule Extraction From Trained Artificial Neural Networks Workshop, AISB96*, Brighton UK, Kwiecień 1996.

[6] S. Bajcar, L. Grzegorczyk. Endangerment by skin cancer among population of south-east part of Poland. Raport instytutowy, Hospital No. 1, Rzeszów, 1997.

[7] R. E. Bellman, L. A. Zadeh. Decision making in a fuzzy environment. *Management Science*, 17:141–164, 1970.

[8] K. P. Bennett, J. A. Blue. A support vector machine approach to decision trees. RPI Math Report 97-100, Rensselaer Polytechnic Institute, Troy, NY, 1997.

[9] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[10] E. Blanzieri, F. Ricci. Advanced metrics for class-driven similarity search. *Proceedings of the International Workshop on Similarity Search*, Firenze, Italy, Wrzesień 1999.

[11] L. Bobrowski. Design of piecewise linear classifiers from formal neurons by some basis exchange technique. *Pattern Recognition*, 24(9):863–870, 1991.

[12] L. Bobrowski. Piecewise–linear classifiers, formal neurons and separability of learning sets. *Proceedings of ICPR*, strony 224–228, 1996.

[13] L. Bobrowski. Generowanie sieci neuropodobnych oraz drzew decyzyjnych w oparciu o kryterium dipolowe. *Symulacja w badaniach i rozwoju*, Jelenia Góra, 1997.

[14] L. Bobrowski. Strategie projektowania sieci neuronalnych. W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, redaktorzy, *Biocybernetyka 2000, Tom 6: Sieci neuronowe*, rozdział I.9, strony 295–321. Akademicka Oficyna Wydawnicza EXIT, 2000.

[15] L. Bobrowski, M. Krętowska, M. Krętowski. Design of neural classifying networks by using dipolar criterions. *Third Conference on Neural Networks and Their Applications*, Kule, Poland, Październik 1997.

[16] L. Bobrowski, M. Krętowski. Induction of multivariate decision trees by using dipolar criteria. D. A. Zighed, J. Komorowski, J. M. Żytkow, redaktorzy, *Principles of data mining and knowledge discovery: 5th European Conference: PKDD'2000*, strony 331–336, Berlin, 2000. Springer Verlag.

[17] A. F. Bowers, C. Giraud-Carrier, J. W. Lloyd. A unifying view of knowledge representation for inductive learning, 2000.

[18] S. Brandt. *Analiza Danych*. Wydawnictwo Naukowe PWN, Warszawa, 1998. tłum. L. Szymanowski.

[19] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[20] L. Breiman. Bias, variance, and arcing classifiers. Raport instytutowy 460, Statistics Department, University of California, 1996.

[21] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[22] L. Breiman. Bias-variance, regularization, instability and stabilization. C. M. Bishop, redaktor, *Neural Networks and Machine Learning*, strony 27–56. 1998.

[23] L. Breiman, J. H. Friedman, A. Olshen, C. J. Stone. *Classification and regression trees*. Wadsworth, Belmont, CA, 1984.

[24] C. E. Brodley, P. E. Utgoff. Multivariate versus univariate decision trees. Raport instytutowy 92-8, Department of Computer Science, University of Massachusetts, Styczeń 1992.

[25] C. E. Brodley, P. E. Utgoff. Multivariate versus univariate decision trees. Raport instytutowy 92-8, Department of Computer Science, University of Massachusetts, Styczeń 1992.

[26] C. Browne, I. Düntsch, G. Gediga. Iris revisited: A comparison of discriminant and enhanced rough set data analysis. L. Polkowski, A. Skowron, redaktorzy, *Rough sets in knowledge discovery*, wolumen 2, strony 345–368. Physica Verlag, Heidelberg, 1998.

[27] I. Bruha, P. Berka. Discretization and fuzzification of numerical attributes in attribute-based learning. P. S. Szczepaniak, P. J. G. Lisboa, J. Kacprzyk, redaktorzy, *Fuzzy Systems in Medicine*, wolumen 41 serii *Studies in Fuzziness and Soft Computing*, strony 112–138. Physica-Verlag (Springer), Heidelberg, 2000.

[28] W. Buntine. Learning classification trees. D. J. Hand, redaktor, *Artificial Intelligence frontiers in statistics*, strony 182–201. Chapman & Hall,London, 1993.

[29] B. Cestnik. Estimating probabilities: A crucial task in machine learning. *Proceedings of the Ninth European Conference on Artificial Intelligence*, strony 147–149, 1990.

[30] V. Cherkassky, F. Mulier. *Learning from data*. Adaptive and learning systems for signal processing, communications and control. John Wiley & Sons, Inc., New York, 1998.

[31] S. L. Chiu. An efficient method for extracting fuzzy classification rules from high dimensional data. *Journal of Advanced Computational Intelligence*, 1(1):31–36, 1997.

[32] P. Clark, R. Boswell. Rule induction with CN2: Some recent improvements. Ives Kodratoff, redaktor, *Proceedings of the Fifth European Conference (EWSL-91)*, Machine Learning, strony 151–163, 1991.

[33] P. Clark, T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[34] I. Colombet, A. Ruelland, G. Chatellier, F. Gueyffier, P. Degoulet, M.-C. Jaulent. Models to predict cardiovascular risk: Comparison of CART, multilayer perceptron and logistic regression. *American Medical Informatics Association Annual Symposium*, 2000.

[35] M. W. Craven. *Extracting comprehensible models from trained neural networks*. Praca doktorska, University of Wisconsin – Madison, 1996.

[36] M. W. Craven, J. W. Shavlik. Using sampling and queries to extract rules from trained neural networks. *Proc. of the Eleventh Int. Conference on Machine Learning*, strony 37–45, New Brunswick, NJ, 1994. Morgan Kaufmann.

[37] J. M. P. da Gama. *Combining Classification Algorithms*. Praca doktorska, Universidade do Porto, 1999.

[38] J. P. Marques de Sá. *Pattern Recognition. Concepts, Methods and Applications*. Springer Verlag, 2001.

[39] T. G. Dietterich. Statistical tests for comparing supervised classification learning algorithms. Raport instytutowy, Department of Computer Science, Oregon State University, Corvallis, OR, 1996.

[40] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[41] W. Duch. Floating gaussian mapping: a new model of adaptive systems. *Neural Network World*, 4:645–654, 1994.

[42] W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:211–219, 1998.

[43] W. Duch, R. Adamczak, K. Grąbczewski. Methodology of extraction, optimization and application of logical rules. *Proceedings of IIS'99*, strony 22–31, Ustroń, Poland, 1999.

[44] W. Duch, R. Adamczak, K. Grąbczewski. Neural optimization of linguistic variables and membership functions. *Proceedings of ICONIP'99*, Perth, Australia, 1999.

[45] W. Duch, R. Adamczak, K. Grąbczewski. Methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.

[46] W. Duch, R. Adamczak, K. Grąbczewski, M. Ishikawa, H. Ueda. Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches. *Proc. of the European Symposium on Artificial Neural Networks (ESANN'97)*, strony 109–114, Bruge, 1997.

[47] W. Duch, R. Adamczak, N. Jankowski. New developments in the feature space mapping model. Raport instytutowy CIL-KMK-2/97, Computational Intelligence Lab, DCM NCU, Toruń, Poland, Październik 1997. (long version).

[48] W. Duch, J. Biesiada, T. Winiarski, K. Grudziński, K. Grąbczewski. Feature selection based on information theory filters and feature elimination wrapper methods. *Proceedings of the International Conference on Neural Networks and Soft Computing (ICNNSC 2002)*, Advances in Soft Computing, strony 173–176, Zakopane, 2002. Physica-Verlag (Springer).

[49] W. Duch, J. Biesiada, T. Winiarski, A. Kachel. Feature selection based on information theory filters and feature elimination wrapper methods. 2003. submitted to the International Conference on Artificial Neural Networks (ICANN 2003).

[50] W. Duch, G. H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.

[51] W. Duch, K. Grąbczewski. Searching for optimal MLP. *Proceedings of the Fourth Conference on Neural Networks and Their Applications*, strony 65–70, Zakopane, Maj 1999.

[52] W. Duch, K. Grąbczewski, R. Adamczak, K. Grudziński, Z. S. Hippe. Rules for melanoma skin cancer diagnosis. *KOSYR 2001*, strony 59–68, Wrocław, 2001.

[53] W Duch, K. Grąbczewski. Heterogeneous adaptive systems. *Proceedings of the World Congress of Computational Intelligence*, Honolulu, Maj 2002.

[54] W. Duch, K. Grudziński, G. Stawski. Symbolic features in neural networks. *Proceedings of the 5th Conference on Neural Networks and Their Applications*, strony 180–185, Zakopane, Poland, Czerwiec 2000.

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

[56] W. Duch, J. Korczak. Optimization and global minimization methods suitable for neural networks. Raport instytutowy CIL-KMK/LSIIT-2/97, 1997.

[57] R. O. Duda, P. E. Hart, D. G. Stork. *Patter Classification*. John Wiley and Sons, New York, 2001.

[58] I. Düntsch, G. Gediga. The rough set engine GROBIAN. *Proceedings of the 15th IMACS World Congress*, wolumen 4, Berlin, Sierpień 1997.

[59] J. P. Egan. Signal detection theory and ROC analysis. *Academic Press*, 1975.

[60] T. Fawcett. Using rule sets to maximize ROC performance. *ICDM*, strony 131–138, 2001.

[61] M. Fernández, C. Hernández. Neural networks input selection by using the training set. *IEEE World Congress on Computational Intelligence. Proceeding of the 1999 IEEE International Joint Conference on Neural Networks*, number 0411, Washington D.C., USA, 1999.

[62] R. A. Fisher. The use of multiple measurements in taxonomic problems. 1936. Reprinted in *Contributions to Mathematical Statistics*, John Wiley & Sons, New York, 1950.

[63] G. L. Foresti, C. Micheloni. Generalized neural trees for pattern recognition. *IEEE Transactions on Neural Networks*, 13(6):1540–1547, 2002.

[64] Y. Freund, R. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, strony 148–156, 1996.

[65] Y. Freund, R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[66] L. M. Fu. Rule learning by searching on adapted nets. *Proceedings of the Ninth National Conference on Artificial Intelligence*, strony 590–595, Anaheim CA, 1991.

[67] L. M. Fu. Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man and Cybernetics*, 23:173–182, 1993.

[68] L. M. Fu. *Neural networks in computer intelligence*. McGraw Hill, New York, 1994.

[69] L. M. Fu. Rule generation from neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 28:1114–1124, 1994.

[70] S. Gallant. *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA, 1993.

[71] P. Geczy, S. Usui. Rule extraction from trained neural networks. *Int. Conf. on Neural Information Processing*, wolumen 2, strony 835–838, New Zealand, Listopad 1997.

[72] R. M. Goodman, P. Smyth. An information theoretic model for rule-based expert systems. International Symposium on Information Theory, 1988. Kobe, Japan.

[73] R. M. Goodman, P. Smyth. The induction of probabilistic rule sets - the ITRULE algorithm. B. Spatz, redaktor, *Proceedings of the sixth international workshop on machine learning*, strony 129–132, San Mateo, CA, 1989. Morgan Kaufmann.

[74] R. M. Goodman, P. Smyth. Information theoretic rule induction. *Proceedings of the 1988 Conference on AI*, London, 1989. Pitman Publishing.

[75] K. Grąbczewski, W. Duch, R. Adamczak. Neuronowe metody odkrywania wiedzy w danych. W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, redaktorzy, *Biocybernetyka 2000, Tom 6: Sieci neuronowe*, rozdział III.20, strony 637–662. Akademicka Oficyna Wydawnicza EXIT, 2000.

[76] K. Grąbczewski, N. Jankowski. Symbolic data transformations for continuous data oriented models. 2003. submitted to ICANN 2003 conference.

[77] K. Grąbczewski, W. Duch. A general purpose separability criterion for classification systems. *Proceedings of the 4th Conference on Neural Networks and Their Applications*, strony 203–208, Zakopane, Poland, Czerwiec 1999.

[78] K. Grąbczewski, W. Duch. The separability of split value criterion. *Proceedings of the 5th Conference on Neural Networks and Their Applications*, strony 201–208, Zakopane, Poland, Czerwiec 2000.

[79] K. Grudziński, W. Duch. SBL-PM: A simple algorithm for selection of reference instances for similarity based methods. *Proceedings of Intelligent*

Information Systems IIS'2000, strony 99–108. Physica Verlag, Springer, 2000.

[80] J. W. Grzymała-Busse. LERS - a system for learning from examples based on rough sets. R. Słowiński, redaktor, *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory*, strony 3–18. Kluwer Academic Publishers, Norwell (MA), 1992.

[81] J. W. Grzymała-Busse. LERS - a knowledge discovery system. R. Słowiński, redaktor, *Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems*, strony 562–565. Physica-Verlag, Heidelberg, 1999.

[82] S. K. Halgamuge, M. Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65:1–12, 1994.

[83] J. A. Hanley, B. J. McNeil. The meaning and use of the area under a receiver operator characteristic (ROC) curve. *Radiology*, 143(1):29–36, Kwiecień 1982.

[84] J. A. Hanley, B. J. McNeil. A method of comparing the areas under receiver operator characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, Wrzesień 1983.

[85] Y. Hayashi. A neural expert system with automated extraction of fuzzy if-then rules. R. Lippmann, J. Moody, D. Touretzky, redaktorzy, *Advances in Neural Information Processing Systems*, wolumen 3. Morgan Kaufmann, San Mateo, CA, 1991.

[86] R. Hayward, C. Ho-Stuart, J. Diederich, E. Pop. RULENEG: extracting rules from a trained ANN by stepwise negation. Raport instytutowy, QUT NRC, Styczeń 1996.

[87] M. J. Healy, T. P. Caudell. Acquiring rule sets as a product of learning in a logical neural architecture. *IEEE Trans. Neural Networks*, 8:461–474, 1997.

[88] Z. S. Hippe. Machine learning - a promising strategy for business information processing? W. Abramowicz, redaktor, *Business Information Systems'97, Academy of Economy Edit. Office*, strony 603–622. Poznań, 1997.

[89] Z. S. Hippe, T. M. Hippe. An attempt to automatize modeling of medical data. *Computers in Medicine*, strony 24–31. Polish Society of Medical Informatics, Łódź, 1997.

[90] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[91] L. Ingber. Very fast simulated re-annealing. *Mathl. Comput. Modelling*, 12:967–973, 1989.

[92] L. Ingber. Simulated annealing: Practice versus theory. *Mathl. Comput. Modelling*, 18(11):29–57, 1993.

[93] L. Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics, Simulated Annealing Applied to Combinatorial Optimization*, 1995.

[94] M. Ishikawa. Rule extraction by succesive regularization. *Proc. of IEEE Int. Conf. on Neural Networks*, strony 1139–1143, Washington, 1996.

[95] I. Jagielska, C. Matthews, T. Whitfort. The application of neural networks, fuzzy logic, genetic algorithms and rough sets to automated knowledge acquisition. *Proceedings of the 4th Int. Conf. on Soft Computing IIZUKA'96*, wolumen 2, strony 565–569, Iizuka, Japan, 1996.

[96] J-S. R. Jang, C. T. Sun. Functional equivalence between radial basis function neural networks and fuzzy inference systems. *IEEE Trans. on Neural Networks*, 4(1):156–158, 1993.

[97] C. Z. Janikow. A genetic algorithm method for optimizing fuzzy decision trees. *Information Sciences*, 89:275–296, 1996.

[98] N. Jankowski, V. Kadirkamanathan. Statistical control of RBF-like networks for classification. *7th International Conference on Artificial Neural Networks*, strony 385–390, Lausanne, Switzerland, Październiernik 1997. Springer-Verlag.

[99] G. H. John, P. Langley. Estimating continuous distributions in bayesian classifiers. *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, San Mateo, 1995. Morgan Kaufmann Publishers.

[100] J. Kacprzyk. Fuzzy sets and fuzzy systems: A brief introduction. P. S. Szczepaniak, P. J. G. Lisboa, J. Kacprzyk, redaktorzy, *Fuzzy Systems in Medicine*, wolumen 41 serii *Studies in Fuzziness and Soft Computing*, strony 3–30. Physica-Verlag (Springer), Heidelberg, 2000.

[101] J. Kacprzyk. *Wieloetapowe sterowanie rozmyte*. Wydawnictwa Naukowo-Techniczne, Warszawa, 2001.

[102] L. Kanal, V. Kumar. *Search in Artificial Intelligence*. Springer Verlag, 1988.

[103] N. Kasabov. Connectionist methods for fuzzy rules extraction, reasoning and adaptation. *Proceedings of the International Conference on Fuzzy Systems, Neural Networks and Soft Computing*, strony 74–77, Iizuka, Japan, 1996. World Scientific.

[104] N. Kasabov. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. The MIT Press, 1996.

[105] N. Kasabov, R. Kozma, W. Duch. Rule extraction from linguistic rule networks and from fuzzy neural networks: Propositional versus fuzzy rules. *Fourth Int. Conf. on Neural Networks and their Applications*, strony 403–406, Marseille, France, 1998.

[106] M. Ait Kbir, K. Maalmi, R. Benslimane, H. Benkirane. Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules. *Pattern Recognition Letters*, 21(6-7):503–509, 2000.

[107] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

[108] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

[109] M. V. Kiselev. Polyanalyst - a machine discovery system inferring functional programs. *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, strony 237–249, Seattle, 1994.

[110] R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs*. Praca doktorska, Stanford University, 1995.

[111] R. Kohavi, B. Becker, D. Sommerfield. Improving simple bayes. *Proceedings of the European Conference on Machine Learning*, 1997.

[112] R. Kohavi, C. Kunz. Option decision trees with majority votes. *Proceedings of the Fourteenth International Conference on Machine Learning*, strony 161–169, 1997.

[113] R. Kohavi, D. Sommerfield, J. Dougherty. Data mining using MLC++: A machine learning library in C++. *Tools with Artificial Intelligence*, strony 234–245. IEEE Computer Society Press, 1996. http://www.sgi.com/tech/mlc.

[114] J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron. Rough sets: A tutorial. S. K. Pal, A. Skowron, redaktorzy, *Rough Fuzzy Hybridization – A New Trend in Decision Making*, strony 3–98. Springer-Verlag, 1999.

[115] B. Kosko. Fuzzy systems as universal approximators. *Proceedings of the First IEEE Conference on Fuzzy Systems*, strony 1153–1162, San Diego, CA, 1992.

[116] S. C. Kremer, D. A. Stacey. NIPS 2000 unlabeled data competition and workshop. http://q.cis.uoguelph.ca/∼skremer/NIPS2000/, 2000.

[117] M. Kurzyński. *Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne*. Prace Naukowe Instytutu Sterowania i Techniki Systemów Politechniki Wrocławskiej. Wydawnictwo Politechniki Wrocławskiej, 1987.

[118] T.-S. Lim, W.-Y. Loh, Y.-S. Shih. An empirical comparison of decision trees and other classification methods. Raport instytutowy, Department of Statistics, University of Wisconsin, Madison, Czerwiec 1997.

[119] T.-S. Lim, W.-Y. Loh, Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000.

[120] W.-Y. Loh, Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

[121] W.-Y. Loh, N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis (with discussion). *Journal of the American Statistical Association*, 83:715–728, 1988.

[122] R. Lowry. Concepts and applications of inferential statistics. http://faculty.vassar.edu/lowry/webtext.html, 1999–2002.

[123] D. D. Margineantu, T. G. Dietterich. Improved class probability estimates from decision tree models. *To appear in Lecture Notes in Statistics*, 2002.

[124] C. McMillan, M. C. Mozer, P. Smolensky. Rule induction through integrated symbolic and subsymbolic processing. J. Moody, S. Hanson, R. Lippmann, redaktorzy, *Advances in Neural Information Processing Systems*, wolumen 4. Morgan Kaufmann, San Mateo, CA, 1992.

[125] M. Mehta, J. Rissanen, R. Agraval. MDL-based decision tree pruning. U.M Fayyad, R. Uthurusamy, redaktorzy, *Proceedings of the First International*

Conference on Knowledge Discovery and Data Mining, strony 216–221, Menlo Park, CA, 1995. AAAI Press.

[126] C. J. Merz, P. M. Murphy. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[127] C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8:283–298, 1978.

[128] R. S. Michalski, K. Kaufman, J. Wnek. The AQ family of learning programs: A review of recent developments and an exemplary application. Reports of the Machine Learning and Inference Laboratory MLI 91-11, School of Information Technology and Engineering, George Mason University, Fairfax, VA, Grudzień 1991.

[129] R. S. Michalski, I. Mozetic, J. Hong, N. Lavrac. Multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proceedings of the Fifth National Conference on Artificial Intelligence*, strony 1041–1045, Philadelphia, PA, 1986. Morgan Kaufmann.

[130] D. Michie, D. J. Spiegelhalter, C. C. Taylor. *Machine learning, neural and statistical classification*. Elis Horwood, London, 1994.

[131] T. Mitchell. *Machine learning*. McGraw Hill, 1997.

[132] J. J. Mulawka. *Systemy Ekspertowe*. Wydawnictwa Naukowo-Techniczne, Warszawa, 1996.

[133] W. Müller, F. Wysotzki. Automatic construction of decision trees for classification. *Annals of Operations Research*, 52:231–247, 1994.

[134] W. Müller, F. Wysotzki. The decision-tree algorithm CAL5 based on a statistical approach to its splitting algorithm. *Machine Learning and Statistics: The Interface.*, strony 45–65, 1997.

[135] S. K. Murthy. *On Growing Better Decision Trees from Data*. Praca doktorska, The Johns Hopkins University, Baltimore, Maryland, 1997.

[136] S. K. Murthy, S. Kasif, S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, Sierpień 1994.

[137] D. Nauck, R. Kruse. Designing neuro-fuzzy systems through backpropagation. W. Pedrycz, redaktor, *Fuzzy Modelling: Paradigms and Practice*, strony 203–228. Kluwer, Boston, 1996.

[138] D. Nauck, R. Kruse. New learning strategies for NEFCLASS. *Proceedings of the Seventh International Fuzzy Systems Association World Congress IFSA'97*, wolumen IV, strony 50–55, Prague, 1997.

[139] D. Nauck, U. Nauck, R. Kruse. Generating classification rules with the neuro-fuzzy system NEFCLASS. *Proceedings of Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS'96)*, Berkeley, 1996.

[140] A. Niederliński. *Regułowe Systemy Ekspertowe*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2000.

[141] A. Ohrn, J. Komorowski. ROSETTA: A rough set toolkit for analysis of data. *Proceedings of the Third International Joint Conference on Information Sciences, Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97)*, wolumen 3, strony 403–407, Durham, NC, USA, Marzec 1997.

[142] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[143] Z. Pawlak. *Rough sets - theoretical aspects of reasoning about data*. Kluver Academic Publishers, 1991.

[144] Z. Pawlak, A. Skowron. Rough membership functions. R. R. Yaeger, M. Fedrizzi, J. Kacprzyk, redaktorzy, *Advances in the Dempster Shafer Theory of Evidence*, strony 251–271. John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1994.

[145] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. B. Schölkopf, C. J. C. Burges, A. J. Smola, redaktorzy, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA., 1998.

[146] pod.red. D. Rogalskiej. *Programiwanie liniowe, Algorytmy i zadania*. Wydawnictwo Uniwersytetu Łódzkiego, Łódź, 1998.

[147] E. Pop, R. Hayward, J. Diederich. RULENEG: extracting rules from a trained ANN by stepwise negation. Raport instytutowy, QUT NRC, Grudzień 1994.

[148] F. Provost, P. Domingos. Well-trained PETs: Improving probability estimation trees. Raport instytutowy IS-00-04, Stern School of Business, New York University, 2000.

[149] F. Provost, T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, Marzec 2001.

[150] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[151] J. Quinlan. Programs for machine learning, 1993.

[152] J. R. Quinlan. Bagging, boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96*, wolumen 1, strony 725–730, Portland, Oregon, Sierpień 1996. AAAI Press / The MIT Press.

[153] J. R. Quinlan, R. M. Cameron-Jones. Oversearching and layered search in empirical learning. *IJCAI*, strony 1019–1024, 1995.

[154] J. R. Quinlan, R. Rivest. Inferring decision trees using the minimum description length principle, 1989.

[155] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:445–471, 1978.

[156] H. J. Runka. *Programowanie Matematyczne, Część I. Programowanie Liniowe*. Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań, 1997.

[157] H. J. Runka. *Programowanie Matematyczne, Część II. Programowanie Nieliniowe*. Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań, 1997.

[158] K. Saito, R. Nakano. Medical diagnostic expert system based on PDP model. *Proc. of IEEE Int. Conf. on Neural Networks*, wolumen 1, strony 255–262, San Diego CA, 1988.

[159] R. Schalkoff. *Pattern Recognition: statistical, structural and neural approaches*. Wiley, 1992.

[160] W. Schiffman, M. Joost, R. Werner. Comparison of optimized backpropagation algorithms. *Proc. of European Symposium on Artificial Neural Networks*, strony 97–104, Brussels, 1993. De facto Publications.

[161] M. J. J. Scott, M. Niranjan, R. W. Prager. Realisable classifiers: Improving operating performance on variable cost problems. *British Machine Vision Conference*, Wrzesień 1998.

[162] S. Sestito, T. Dillon. *Automated knowledge acquisition*. Prentice Hall, Australia, 1994.

[163] I. K. Sethi, J. H. Yoo. Symbolic approximation of feedforward neural networks. E. S. Gelsema, L. N. Kanal, redaktorzy, *Pattern Recognition in Practice*, wolumen 4. North-Holland, New York, 1994.

[164] R. Setiono, H. Liu. Understanding neural networks via rule extraction. *Proc. of the 14th Int. Joint Conference on Artificial Intelligence*, strony 480–485, Montreal, Quebec, 1995. Morgan Kaufmann.

[165] N. Shang, L. Breiman. Distributioon based trees are more accurate. *Proceedings of ICONIP'96*, wolumen 1, strony 133–138, 1996.

[166] P. Smyth, R. M. Goodman. Rule induction using information theory. G. Piatetsky-Shapiro, W. J. Frawley, redaktorzy, *Knowledge Discovery in Databases*, strony 159–176. The MIT Press, Cambridge, 1991.

[167] P. Smyth, R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, Sierpień 1992.

[168] J. Stefanowski. *Algorytmy indukcji reguł decyzyjnych w odkrywaniu wiedzy*, wolumen 261. Wydawnictwo Politechniki Poznańskiej, 2001.

[169] B. Šter, A. Dobnikar. Neural networks in medical diagnosis: Comparison with other methods. A. B. Bulsari et al., redaktor, *Proceedings of the International Conference EANN '96*, strony 427–430, 1996.

[170] M. M. Sysło, N. Deo, J. S. Kowalik. *Algorytmy optymalizacji dyskretnej*. Wydawnictwo Naukowe PWN, Warszawa, 1995.

[171] R. Tadeusiewicz, A. Izworski, J. Majewski. *Biometria*. Wydawnictwa AGH, Kraków, 1993.

[172] A-H. Tan. Rule learning and extraction with self-organizing neural networks. *Proc. of the 1993 Connectionist Models Summer School*, strony 192–199, Hillsdale, NJ, 1994. Lawrence Erlbaum Associates.

[173] M. Tan, L. Eshelman. Using weighted networks to represent classification knowledge in noisy domains. *Proceedings of the Fifth International Conference on Machine Learning*, strony 121–134, Ann Arbor, MI, 1988.

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

[175] A. B. Tickle, R. Andrews, M. Golea, J. Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Trans. Neural Networks*, 9:1057–1068, 1998.

[176] A. B. Tickle, M. Orlowski, J. Diederich. DEDEC: decision detection by rule extraction from neural networks. Raport instytutowy, QUT NRC, Wrzesień 1994.

[177] G. Towell, J. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

[178] A. Ultsch. Knowledge extraction from self-organizing neural networks. O. Opitz, B. Lausen, R. Klar, redaktorzy, *Information and Classification*, strony 301–306. Springer, Berlin, 1993.

[179] J. M. Żurada, A. Łozowski. Generating linguistic rules from data using neuro-fuzzy framework. *4th Int. Conf. on Soft Computing IIZUKA'96*, wolumen 2, strony 618–621, Iizuka, Japan, 1996.

[180] P. E. Utgoff, C. E. Brodley. Linear machine decision trees. Raport instytutowy UM-CS-1991-010, Department of Computer Science, University of Massachusetts, , 1991.

[181] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[182] S. M. Weiss, I. Kapouleas. 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.

[183] D. R. Wilson, T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 11:1–34, 1997.

[184] P. H. Winston. *Artificial Intelligence*. Addison Wesley, 1992.

[185] J. Wnek, K. Kaufman, E. Bloedorn, R. S. Michalski. Inductive learning system AQ15c: The method and user's guide. Reports of the Machine

Learning and Inference Laboratory MLI 95-4, George Mason University, Fairfax, VA, Marzec 1995.

[186] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[187] B. Zadrozny, C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. *Proc. 18th International Conf. on Machine Learning*, strony 609–616. Morgan Kaufmann, San Francisco, CA, 2001.

[188] F. Zarndt. A comprehensive case study: An examination of machine learning and connectionist algorithms. Praca magisterska, Dept. of Computer Science, Brigham Young University, 1995.