Politechnika Łódzka
Wydział Elektroniki i Elektrotechniki
Katedra Informatyki Stosowanej

Rozprawa doktorska

Konstrukcja klasyfikatorów minimalnoodległościowych o strukturze sieciowej

SZYMON GRABOWSKI

Promotor: prof. dr hab. inż. Dominik Sankowski

Łódź, 2003 r.

Spis treści

Wykaz skrótów i oznaczeń stosowanych w rozprawie

1-NN	– reguła decyzyjna jeden najbliższy sąsiad (ang. *nearest neighbor*)
1R	– system reguł oparty na drzewach decyzyjnych jednego poziomu (ang. *1-rules*)
A-NNS	– wersja przybliżona zagadnienia szybkiego szukania najbliższego sąsiada (ang. *approximate nearest neighbor search*)
BD-SS	– strategia dwukierunkowej zmiany zestawu cechy (ang. *bidirectional selection strategy*)
BSS	– strategia kolejnego usuwania cech (ang. *backward selection strategy*)
CNN	– algorytm redukcji zbioru odniesienia „skondensowany najbliższy sąsiad” (ang. *condensed nearest neighbor*)
ECOC	– schemat dekompozycyjny z wyjściowymi kodami samokorygującymi (ang. *error-correcting output codes*)
FSS	– strategia kolejnego dodawania cech (ang. *forward selection strategy*)
GG	– graf Gabriela (ang. *Gabriel graph*)
GGN	– klasyfikator indukowany grafem Gabriela
G–K	– algorytm redukcji Gowdy–Krishny
J–V	– Jóźwik–Vernazza; alternatywne określenie schematu dekompozycji zadania wielodecyzyjnego PWC zastosowanego w klasyfikatorach minimalnoodległościowych
k-NCN	– reguła decyzyjna *k* scentrowanych sąsiadów (ang. *k nearest centroid neighbors*)
k-NN	– reguła decyzyjna *k* najbliższych sąsiadów (ang. *k nearest neighbors*)
k-NSN	– reguła decyzyjna *k near surrounding neighbors*
MFS	– klasyfikator „wiele podzbiorów cech” (ang. *multiple feature subsets*)
M–M	– Moreira–Mayoraz; alternatywne określenie schematu dekompozycji zadania wielodecyzyjnego PWC-CC
NBC	– naiwny klasyfikator Bayesa (ang. *naive Bayes classifier*)
NNFP	– reguła najbliższego sąsiada z projekcją cech (ang. *nearest neighbor with feature projection*)
NNS	– problem szybkiego szukania najbliższego sąsiada (ang. *nearest neighbor search*)
NS	– najbliższy sąsiad

OPC	– schemat dekompozycji zadania wielodecyzyjnego „jedna klasa przeciwko pozostałym” (ang. *one-per-class*)
PP	– partycjonowanie przestrzeni w schemacie z lokalnym wyborem zbioru zredukowanego
PWC	– schemat dekompozycji zadania wielodecyzyjnego oparty na zestawieniu par klas (ang. *pairwise coupling*)
PWC-CC	– schemat dekompozycji zadania wielodecyzyjnego „skorygowany zestaw par klas” (ang. *pairwise coupling — correcting classifiers*)
RMHC	– „wspinaczka po wzgórzu z losowymi mutacjami”, stochastyczna technika uczenia (ang. *random mutation hill climbing*)
RNG	– graf pokrewieństwa (ang. *relative neighborhood graph*)
RNGN	– klasyfikator indukowany grafem pokrewieństwa
SNN	– algorytm selektywny redukcji zbioru odniesienia (ang. *selective nearest neighbor*)
SNN-CR	– modyfikacja algorytmu selektywnego redukcji zbioru odniesienia z rozstrzyganiem sytuacji oryginalnie nieokreślonych (ang. *selective nearest neighbor — careful removal*)
UCI	– Uniwersytet Kalifornijski w Irvine (University of California, Irvine)
wNNFP	– ważona reguła najbliższego sąsiada z projekcją cech (ang. *weighted nearest neighbor with feature projection*)

$\forall$	– kwantyfikator ogólny
δ_{ij}	– symbol Kroneckera
$\lambda(\omega_i \mid \omega_j)$	– funkcja kosztu
μ	– średnia odległość między losową parą obiektów w zbiorze
σ	– odchylenie standardowe histogramu odległości między wszystkimi parami próbek w zbiorze
θ	– miara trudności (ang. *measure of difficulty*) dla korelacji zespołu klasyfikatorów

acc	– jakość (trafność predykcji) klasyfikatora (ang. *accuracy*)
c	– liczba klas
d	– wymiar przestrzeni
$d(x, y)$	– odległość między obiektami x i y
$d_p(x, y)$	– odległość Minkowskiego między obiektami x i y
D	– miara niezgodności dla korelacji pary klasyfikatorów

D_i – klasyfikator składowy (komponent) w zespole

E – wartość oczekiwana zmiennej losowej

f – funkcja gęstości rozkładu prawdopodobieństwa

$\hat{f}$ – przybliżona (estymowana) funkcja gęstości rozkładu prawdopodobieństwa

f_i – funkcja binarna skojarzona z *i*-tą klasą w schemacie dekompozycyjnym „jedna klasa przeciwko pozostałym"

h – liczność zbioru zredukowanego

K_B – reguła decyzyjna Bayesa, klasyfikator Bayesa

l_j – etykieta klasy

L – zbiór etykiet klas

m_1, m_2 – liczba mutacji w odmianach algorytmu redukcji Skalaka

n – liczność zbioru

p^* – prawdopodobieństwo mylnej klasyfikacji przez klasyfikator Bayesa

$p(e)$ – prawdopodobieństwo mylnej klasyfikacji przez regułę decyzyjną 1-NN;

$p(x)$ – łączny rozkład gęstości prawdopodobieństwa wystąpienia wektora x

$p(\omega_j)$ – prawdopodobieństwo *a priori* klasy ω_j

$p(x|\omega_j)$ – funkcja gęstości prawdopodobieństwa klasy ω_j

$p(\omega_j|x)$ – prawdopodobieństwo, że wektor x należy do klasy ω_j, prawdopodobieństwo *a posteriori* klasy ω_j

P – zbiór odniesienia w problemie szybkiego szukania najbliższego sąsiada

q – próbka testowa

Q – współczynnik Yule'a

r – promień szukania w problemie szybkiego szukania najbliższego sąsiada

R – zbiór liczb rzeczywistych

shr – przesunięcie bitowe w prawo

T – zbiór odniesienia w postaci wektorów cech

X – przestrzeń metryczna

xor – alternatywa wyłączająca

Wykaz ilustracji

Wykaz tabel

Wstęp

Informatyka w dużej mierze polega na czynieniu możliwie dobrego użytku z danych. „Czynienie dobrego użytku" oznacza tu m. in. analizę i kategoryzację danych, transmisję i wyszukiwanie danych oraz wygodne metody ich prezentacji. Rozpoznawanie obrazów podejmuje problemy centralne dla całej informatyki: analizę danych pojmowaną jako znajdowanie trendów i zależności oraz predykcję właściwości nowych danych; ich reprezentację i interpretację; selekcję informacji, czyli odrzucenie danych nieistotnych; wreszcie organizację danych, czyli dostosowanie własności konkretnego algorytmu do zasobów sprzętowych.

Danych przetwarzanych cyfrowo przybywa, gdyż rosną przepustowości kanałów informacyjnych (Internetu, sieci lokalnych, magistral *etc*.), coraz bardziej dostępne są nowoczesne źródła pozyskiwania informacji, np. aparaty cyfrowe, wreszcie: coraz więcej osób w niebanalny sposób korzysta z komputerów i rozumie, iż zagadnienia, którymi zajmują się w pracy czy w domu, często dadzą rozwiązać się przy pomocy metod informatycznych.

Narasta zrozumienie faktu, iż komputer może być czymś więcej niż nowoczesną maszyną do pisania, rozbudowanym kalkulatorem, uniwersalnym magnetowidem czy automatem do gier. Upowszechnienie się w ostatnich latach technologii np. rozpoznawania pisma maszynowego i ręcznego oraz rozpoznawania mowy wynika po części z „brutalnej siły" dzisiejszych procesorów, ale także z coraz doskonalszych algorytmów, zaliczanych do metod rozpoznawania obrazów i dziedzin pokrewnych.

Moc przetwarzania to nie wszystko. Przykładowo, słownik ortograficzny języka polskiego w edytorze tekstu MS Word 2000 oferuje nieco gorszą jakość podpowiedzi niż analogiczny słownik w wersji Word 97.[1] Nasuwa się proste wyjaśnienie tego fenomenu: w wersji 2000 powiększono bazę słów, nie zmieniając przy tym algorytmów przybliżonego dopasowywania. Powiększenie słownika oznaczało dodanie pewnej liczby rzadko występujących słów, które tym samym stosunkowo często (zbyt często w praktyce) pojawiają się na liście podpowiedzi. Dotykamy tu kluczowej kwestii rozpoznawania obrazów: pewne dane są „mniej ważne" od innych, a zatem dobry

[1] S. Deorowicz, korespondencja prywatna, 2002.

algorytm powinien bardziej skupiać się na tym, co jest „esencją” niż na danych mniejszej wagi.

Skupienie się na danych istotnych dotyczy zbiorów skończonych, czyli przypadków realnych. Klasyczna reguła klasyfikacyjna *k* najbliższych sąsiadów (*k*-NN), która jest punktem wyjścia dla większości rozważań zamieszczonych w niniejszej rozprawie, jest asymptotycznie optymalna, co wszak wcale nie musi oznaczać najlepszej predykcji w sytuacjach praktycznych. Spostrzeżenie to było inspiracją do badań nad konstrukcją klasyfikatorów, które — pomimo tego, że ich jakość w przypadku asymptotycznym pozostaje nieznana — w praktyce osiągają często błędy mniejsze niż reguła *k*-NN.

Niniejsza rozprawa prezentuje szereg algorytmów klasyfikacji konkurencyjnych wobec metod podawanych w literaturze pod względem jakości i/lub szybkości klasyfikacji. Wielość zaproponowanych metod wiąże się z różnorakością wymogów dotyczących szybkości klasyfikacji i trafności predykcji w rozmaitych zastosowaniach. Ogólnie biorąc, zagadnieniem, którego dotyczy niniejsza praca, jest klasyfikacja nieparametryczna nadzorowana. Nieparametryczność oznacza, iż w danym zadaniu nie jest znana gęstość rozkładu prawdopodobieństw przynależności punktów przestrzeni cech do klas. Jest to sytuacja bardzo często występująca w praktyce i będąca niemałym wyzwaniem. Słowo „nadzorowana” informuje, że rozpoznawanie obiektów korzysta z tzw. zbioru uczącego, tj. zbioru obiektów ze znanymi etykietami klas.

Mało zbadanym podejściem do konstrukcji klasyfikatorów jest wykorzystanie koncepcji symetrycznego sąsiedztwa, tj. sąsiedztwa uwzględniającego nie tylko bliskość, ale i przestrzenny układ zbioru sąsiadów (O’Callaghan, 1975). Jednym z nielicznych algorytmów tej kategorii jest reguła decyzyjna „*k* scentrowanych sąsiadów” (*k Nearest Centroid Neighbors*, *k*-NCN) (Sánchez i in., 1997a). W niniejszej pracy zaproponowano nową regułę decyzyjną *k Near Surrounding Neighbors* (*k*-NSN), zainspirowaną regułą *k*-NCN i ulepszającą jej sąsiedztwo w wyniku stochastycznej procedury zaliczanej w literaturze do klasy *random mutation hill climbing*. Zaproponowana reguła *k*-NSN zwykle osiąga wyższą jakość predykcji niż *k*-NCN i znacząco wyższą jakość predykcji niż *k*-NN.

Innym — bardzo zresztą w ostatnich latach modnym — kierunkiem poszukiwań klasyfikatorów o możliwie wysokiej zdolności predykcji, jest zastosowanie struktury sieciowej zespołu klasyfikatorów. Topologie sieci klasyfikatorów są rozmaite: od równoległej sieci komunikujących lub nie komunikujących się komponentów, poprzez

klasyfikatory lokalnie wybierające odpowiedni dla danej próbki testowej klasyfikator składowy, aż po klasyfikator kaskadowy (wieloetapowy), w którym „łatwe" próbki klasyfikowane są w krótszym czasie niż próbki „trudne". W pracy zaproponowano schematy dotyczące wszystkich trzech naszkicowanych głównych koncepcji klasyfikatorów o strukturze sieciowej.

Drugim podstawowym aspektem klasyfikacji jest jej szybkość. W przypadku klasyfikatorów minimalnoodległościowych, a zwłaszcza reguły 1-NN, możliwe są dwa, do pewnego stopnia niezależne podejścia: użycie algorytmów szybkiego szukania najbliższego sąsiada (sąsiadów) oraz redukcja zbioru odniesienia. Do obu wymienionych zagadnień niniejsza praca wnosi pewien wkład. Po pierwsze, podano algorytm szukania najbliższego sąsiada w metryce miejskiej w czasie zależnym subliniowo od liczności zbioru odniesienia w najgorszym przypadku. Algorytm ten wykorzystuje pewną własność użytej metryki i cechuje się względną prostotą oraz elastycznością (możliwość użycia współczynnika kompromisu między kosztem przetwarzania wstępnego a czasem szukania). Wysokie koszta wstępnej obróbki są, jak się wydaje, nie do uniknięcia w tej klasie algorytmów. Warto podkreślić, że algorytm ten (przy założeniu korzystnych parametrów wejściowych zbioru) można zastosować w połączeniu z niemal każdym minimalnoodległościowym algorytmem klasyfikacji, zaś minimalizacja czasu w najgorszym przypadku predystynuje go do zastosowań *on-line*.

Odnośnie kwestii redukcji zbioru odniesienia, zaproponowano w rozprawie nowe podejście. Jest nim algorytm lokalnego wyboru zbioru zredukowanego, tj. taki, w którym dana próbka testowa jest klasyfikowana przy użyciu jednego z wielu uprzednio wygenerowanych zbiorów zredukowanych, zaś samo kryterium wyboru bazuje na klasteryzacji lub podziale przestrzeni hiperpłaszczyznami i jest tym samym bardzo szybkie.

Rozprawa niniejsza składa się z siedmiu rozdziałów. Rozdział 1 wprowadza w tematykę rozpoznawania obrazów, ze szczególnym uwzględnieniem klasyfikatorów minimalnoodległościowych. Przedmiotem Rozdziału 2 jest jedno z ważniejszych zagadnień dziedziny — selekcja cech. Rozdział 3 szkicuje główne koncepcje klasyfikatorów o strukturze sieciowej oraz przedstawia wyniki eksperymentów autora pracy z własną modyfikacją klasyfikatora „wiele podzbiorów cech" (*Multiple Feature Subsets*, MFS), będącego równoległym podejściem do problemu selekcji cech. Wyniki te — jak dotychczas, dość wstępne — sugerują wyższość zaproponowanego wariantu MFS nad tradycyjnym, tj. nie opartym na równoległości, podejściem do selekcji cech.

Rozdziały 4 i 5 poświęcone są problemowi szybkiego szukania najbliższego sąsiada. W Rozdziale 4 opisano ten problem, jego warianty i zastosowania, a także dokonano obszernego przeglądu istniejących rozwiązań heurystycznych. W Rozdziale 5 przedstawiono, wraz z wynikami implementacji, wspomniany wyżej nowy algorytm szukania najbliższego sąsiada w metryce miejskiej. Algorytm ten cechuje się elastycznością i stosunkową prostotą. Rozdział 6 kontynuuje rozważania nad aspektem szybkościowym klasyfikacji; tym razem podjęte jest zagadnienie redukcji zbioru odniesienia. W rozdziale tym opisano opracowany przez autora rozprawy algorytm lokalnego wyboru zbioru odniesienia dla reguły 1-NN przewyższający w testach algorytmy tradycyjne (tj. działające z pojedynczym zbiorem odniesienia) tak pod względem trafności predykcji, jak i pod względem szybkości działania. Ponadto w rozdziale zamieszczono obszerne eksperymentalne porównanie zaproponowanego podejścia z algorytmami redukcji podawanymi w literaturze. Ostatni rozdział dotyczy konstrukcji algorytmów o możliwie wysokiej jakości klasyfikacji. Zaprezentowano w nim nowy algorytm *k*-NSN działający w oparciu o koncepcję symetrycznego sąsiedztwa, schemat równoległy z prostym głosowaniem wykorzystujący wiele klasyfikatorów składowych typu „*k* sąsiadów" (tj. *k*-NN, *k*-NCN lub *k*-NSN), z których każdy może działać w oparciu o „własną" wartość parametru *k*, a także rodzinę klasyfikatorów kaskadowych o korzystnych stosunkach jakości predykcji do szybkości klasyfikacji, intensywnie wykorzystującą obie wspomniane właśnie koncepcje autora rozprawy.

Niemal każdy z algorytmów zaprezentowanych w niniejszej rozprawie, stanowiący własne osiągnięcie autora, cechuje się elastycznością, a zatem podatnością na dalsze ulepszenia i modyfikacje, a także możliwością dostosowania jego właściwości do różnorakich praktycznych aplikacji. Zaproponowane algorytmy przetestowano na szeregu rzeczywistych i syntetycznych zbiorów danych. Zbiory te dotyczą takich dziedzin, jak m. in. medycyna, kontrola jakości w przemyśle i detekcja obiektów na zdjęciach lotniczych; większość z nich jest szeroko używana w literaturze przedmiotu. Dokładny opis użytych rzeczywistych zbiorów danych zawiera Załącznik. Większość wyników zaprezentowanych w rozprawie została wcześniej opublikowana przez autora na konferencjach krajowych (SiS'99, SiS'00, KOSYR'01, SiS'01, SiS'02, KOSYR'03) lub zagranicznych (konferencje IEEE: TCSET'02, CADSM'03), w pismach naukowych (*Biocybernetics and Biomedical Engineering*, *Prace Przemysłowego Instytutu Elektroniki*), a także zamieszczono w przygotowywanym do druku rozdziale monografii „Frontiers of Remote Sensing Info Processing" (ed. S. Patt, World Scientific Publishing

Co. Pte. Ltd., Singapur). Ogółem, lista publikacji autora rozprawy liczy 29 pozycji, z czego w rozprawie zacytowanych jest 18 prac, ściśle związanych z przedmiotem doktoratu.

Koncepcja klasyfikatorów minimalnoodległościowych, tj. związanych z regułą k najbliższych sąsiadów i jej odmianami, liczy już ponad pół wieku, a jednak, zdaniem autora niniejszej rozprawy, nie spotkała się do tej pory z takim odbiorem, na jaki zasługuje. Autor wyraża nadzieję, że niniejsza praca choć w niewielkim stopniu przyczyni się do zwiększenia zainteresowania tym kierunkiem, zarówno w aspekcie teoretycznym, jak i aplikacyjnym.

Jak wyżej wspomniano, wszystkie algorytmy zaproponowane w rozprawie zostały zaimplementowane i przetestowane. Ponadto autor rozprawy zaimplementował wszystkie wykorzystywane w porównaniach algorytmy znane z literatury. Naturalnie, żaden naukowiec nie zliczy własnych koncepcji, które okazały się chybione bądź zostały zarzucone. W podobnych bólach rodziła się i ta praca. Większość kodów źródłowych została napisana w języku Delphi i uruchomiona na komputerze PC Celeron 466. Algorytm szybkiego szukania najbliższego sąsiada został napisany w języku C++. Prace związane z implementacją algorytmów wymagały od autora rozprawy nie tylko wieloletniej praktyki programistycznej, ale i znajomości wybranych struktur danych oraz — w ograniczonym zakresie — wiedzy z dziedziny analizy algorytmów.

W tym miejscu chciałbym złożyć wyrazy podziękowania kilku osobom. Dziękuję mojemu promotorowi prof. Dominikowi Sankowskiemu za opiekę nad pracą, dr. Adamowi Jóźwikowi za wprowadzenie mnie w dziedzinę rozpoznawania obrazów, niezliczone wskazówki i interesujące dyskusje, kol. Sebastianowi Deorowiczowi za liczne ciekawe rozmowy, kol. Zofii Stawskiej za wnikliwą lekturę tekstu i szereg cennych uwag oraz dr. Arne Anderssonowi z uniwersytetu w Uppsali w Szwecji za zgodę na zamieszczenie Rys. 1.5 pochodzącego z jego pracy. *Last not least* — dziękuję Żonie i Rodzicom za wsparcie duchowe i wiele cierpliwości, tak potrzebne w tym ciężkim, ale ekscytującym okresie pisania doktoratu.

Cel, zakres i tezy pracy

Celem pracy jest skonstruowanie nowych algorytmów klasyfikacji bezparametrycznej nadzorowanej, należących do rodziny klasyfikatorów minimalnoodległościowych, tj. wywodzących się z reguły *k* najbliższych sąsiadów (*k*-NN). Założono, że funkcja kosztu jest symetryczna, tj. że koszt każdej pomyłki klasyfikatora jest taki sam.

Duży nacisk w pracy położono na aspekt szybkościowy klasyfikacji, czego owocem jest m. in. nowy algorytm szybkiego szukania najbliższego sąsiada w metryce miejskiej. Algorytm ten może być wykorzystany w wielu klasyfikatorach minimalno-odległościowych, co więcej, w naturalny sposób można sterować kompromisem między kosztem pamięciowo-czasowym wstępnej obróbki a kosztem szukania. Algorytm ten cechuje się czasem szukania subliniowym w liczności zbioru w najgorszym przypadku, co umożliwia wykorzystanie go zwłaszcza w zastosowaniach klasyfikacji *on-line*. Użycie metryki miejskiej wypływa z teoretycznych i praktycznych przesłanek, a ponadto jest korzystne dla szybkości działania algorytmu.

Zakres prac obejmował także zagadnienie redukcji zbioru odniesienia dla klasyfikatora typu jeden najbliższy sąsiad (1-NN) oraz poprawę jakości klasyfikacji przy użyciu algorytmów wykorzystujących koncepcję symetrycznego sąsiedztwa oraz przy użyciu klasyfikatorów o strukturze sieciowej (równoległych); idee niniejsze umożliwiły ponadto zaprojektowanie rodziny klasyfikatorów kaskadowych cechujących się korzystną relacją trafności predykcji do szybkości klasyfikacji.

Weryfikację opracowanych algorytmów autor rozprawy przeprowadził korzystając z szeregu rzeczywistych i syntetycznych zbiorów danych. Wykorzystanymi zbiorami rzeczywistymi były:

- szeroko używane w literaturze przedmiotu zbiory Bupa, Glass, Iris, Pima i Wine, dostępne w „Machine Learning Repository" Uniwersytetu Kalifornijskiego w Irvine (Merz i Murphy, 1996);
- zbiory Ferrites, FerritesOld i RB8.PAT, dotyczące kontroli jakości rdzeni ferrytowych, które były produkowane w zakładach „Polfer" w Warszawie (Nieniewski i in., 1999; Jóźwik i in., 1996);

- zbiór Remotes, związany z zadaniem detekcji obiektów (pól upraw) na zdjęciach lotniczych wykonanych w rejonie Feltwell w Wielkiej Brytanii (Roli, 1996). Dane te wykorzystywane były w grancie NATO nr PST.CLG.977258 (2001–2002) dotyczącym zastosowań nieparametrycznych metod rozpoznawania obrazów w aplikacjach *remote sensing*, którego autor niniejszej rozprawy był uczestnikiem (kierownik grantu: prof. C.-H. Chen z N. Dartmouth Coll., MA, USA).

Tezy rozprawy sformułowano następująco:

1. W niskich wymiarach ($d \leq 5$) możliwe jest znajdowanie najbliższego sąsiada w deterministycznym subliniowym czasie w metryce miejskiej.
2. Lokalny wybór zredukowanego zbioru odniesienia prowadzi do osiągnięcia wyższej jakości klasyfikacji niż oferowana przez pojedynczy zbiór zredukowany, zwłaszcza przy bardzo wysokich wymaganiach szybkościowych nałożonych na klasyfikację.
3. Możliwe jest stworzenie równoległej sieci klasyfikatorów typu „*k* sąsiadów", osiągającej wyższą jakość predykcji niż klasyfikator bazowy przy umiarkowanym spowolnieniu klasyfikacji, umożliwiającej ponadto, w połączeniu z koncepcją tzw. symetrycznego sąsiedztwa, projektowanie klasyfikatorów kaskadowych o korzystnych relacjach szybkości do jakości klasyfikacji.

Rozdział I
Podstawy rozpoznawania obrazów

Podstawowymi zagadnieniami dziedziny zwanej rozpoznawaniem obrazów (ang. *pattern recognition*) są regresja i klasyfikacja.

Regresją nazywamy predykcję wartości ciągłej numerycznej zmiennej wyjściowej obiektu na podstawie *d* cech (ang. *features*, *attributes*) wejściowych (Duda i Hart, 1973; Bhattacharyya i Johnson, 1977). Innymi słowy, regresja polega na znalezieniu funkcji możliwie dobrze aproksymującej prawdziwe wyjścia obiektów (jakość estymowana jest przy użyciu pewnej miary, np. średniego błędu kwadratowego, na wyodrębnionym zbiorze testowym).

Klasyfikacja jest to predykcja wartości zmiennej dyskretnej obiektów opisanych *d*-wymiarowymi wektorami cech. Różnica między regresją a klasyfikacją polega więc na tym, że odgadywane wyjścia w pierwszym zagadnieniu są ciągłe, a w drugim dyskretne i na ogół bez wprowadzonej relacji porządku między nimi. Wspomniane wartości dyskretne identyfikowane są z klasami, a ściślej z etykietami (nazwami) klas.

Można wyróżnić pojęcia: klasyfikacja nadzorowana (ang. *supervised classification*) i nienadzorowana (ang. *unsupervised classification*). W pierwszym przypadku etykiety klas nieznanych obiektów odgaduje się na podstawie zbioru obiektów o znanych etykietach; jest to tzw. zbiór uczący (ang. *training set*, *learning set*). Właściwość predykcji etykiet nieznanych próbek na podstawie danego zbioru uczącego nazywa się również zdolnością generalizacji klasyfikatora. W przypadku klasyfikacji nienadzorowanej zbiór uczący nie jest dany. Zadanie polega wówczas na rozdzieleniu zbioru obiektów na dwa lub więcej podzbiorów w taki sposób, aby obiekty w obrębie pojedynczego podzbioru były do siebie możliwie podobne (w przestrzeni zadanych cech i w sensie określonej metryki lub miary podobieństwa). Tak utworzone podzbiory nazywamy często klastrami (ang. *clusters*), a proces wyodrębniania klastrów — klasteryzacją (ang. *clustering*). Konkretnymi przykładami klasteryzacji są np. segmentacja obiektów w obrazach 2- i 3-wymiarowych lub kategoryzacja dokumentów tekstowych np. na potrzeby wyszukiwarek sieciowych.

Przedmiotem niniejszej pracy jest klasyfikacja nadzorowana. Ten rodzaj klasyfikacji ma bardzo liczne zastosowania, by wymienić choćby wspomaganą komputerowo

diagnostykę medyczną, kontrolę jakości artykułów przemysłowych, detekcję obiektów na zdjęciach satelitarnych i lotniczych (*remote sensing*) czy rozpoznawanie pisma maszynowego i ręcznego (*Optical Character Recognition*, OCR).

Należy dodać, iż metody rozwijane w rozprawie dotyczą klasyfikacji nieparametrycznej, tj. dotyczącej przypadku, gdy nie jest znany *a priori* rozkład prawdopodobieństw przynależności obiektów przestrzeni cech do klas. Jedyne informacje umożliwiające wówczas klasyfikację niesione są przez zbiór uczący. Taka sytuacja ma bardzo często miejsce w praktyce i pojawia się wszędzie tam, gdzie brakuje wystarczających podstaw do przyjęcia z góry odpowiedniego modelu probabilistycznego dla rozważanego problemu. W niniejszej pracy założono ponadto, że zbiór uczący jest niezmienny dla danego zadania, tj. w szczególności, w czasie klasyfikacji kolejnych obiektów nie wolno korzystać z żadnej informacji dotyczącej już sklasyfikowanych próbek.

1.1. Wiadomości wstępne

Każdy algorytm klasyfikacji ma swoje obciążenie (ang. *bias*), które można zdefiniować jako „dowolną podstawę dla preferowania tej, a nie innej, generalizacji, za wyjątkiem ścisłej zgodności z obserwowanym zbiorem uczącym" (Mitchell, 1980). Innymi słowy, jedynie reguła, która próbki testowe o wartościach dokładnie pokrywających się z pewnym obiektem (obiektami) zbioru testowego przypisuje do klasy tego obiektu(-ów), a w pozostałych przypadkach powstrzymuje się od decyzji, jest „sprawiedliwa". Każda inna reguła decyzyjna obarczona jest obciążeniem, czyli arbitralnie wybraną „skłonnością" do podejmowania takiej, a nie innej, decyzji. Istnieje tzw. prawo zachowania dla generalizacji (ang. *conservation law for generalization performance*), na mocy którego żadne obciążenie reguły decyzyjnej nie może owocować wyższą średnią jakością predykcji od jakiegokolwiek innego obciążenia (uwzględniając także losowe zgadywanie), jeśli rozpatrujemy wszystkie możliwe problemy (Wolpert, 1993; Schaffer, 1994). Brzmi to skrajnie pesymistycznie, pamiętać jednak trzeba, że w praktyce pewne rozkłady występują znacznie częściej niż inne i dzięki temu prawo zachowania dla generalizacji ważne jest jedynie w teorii.

W rozważaniach nad konstrukcją klasyfikatora często przytacza się zasadę metodologiczną znaną pod nazwą brzytwy Ockhama, którą można sformułować w naszym kontekście następująco (Blumer i in., 1987): „należy zawsze preferować

prostszą z dwóch konkurencyjnych (tj. oferujących podobną jakość na zbiorze uczącym) hipotez". Uzasadnia się to przy pomocy argumentu kombinatorycznego: hipotez prostych jest mniej niż hipotez złożonych, dlatego prawdopodobieństwo, iż hipoteza prostsza dobrze pasuje do zbioru uczącego w wyniku przypadku jest mniejsze niż analogiczne prawdopodobieństwo dla hipotezy bardziej złożonej (Mitchell, 1997). Domingos (1998) przeprowadza dogłębne rozważania nt. interpretacji brzytwy Ockhama, wyciągając wniosek: „jeśli model o niskim błędzie na zbiorze uczącym został znaleziony w wyniku inspekcji dostatecznie małej liczby modeli, to jest prawdopodobne, że będzie on oferował także niski błąd na zbiorze testowym". Zauważmy, iż w świetle tej konkluzji wybrany model klasyfikatora nie musi koniecznie być prosty — wystarczy, jeśli jego znalezienie nie wymagało przeglądu zbyt wielu modeli, a już można praktycznie nie obawiać się znaczących rozbieżności między błędem na zbiorze uczącym a błędem na zbiorze testowym.

1.1.1. Definicja zadania klasyfikacji nadzorowanej

Niech $T = \{x_1, ..., x_n\}$ będzie zbiorem wektorów postaci $x_i = [x_{i,1}, ..., x_{i,d}]$ i niech będzie dane przyporządkowanie:

$$\hat{f} : T \to L, \tag{1.1}$$

gdzie $L = \{1, ..., c\}$ jest zbiorem możliwych etykiet klas. Funkcja $\hat{f}$ zazwyczaj nie jest znana w postaci analitycznej, a tylko jako zbiór par: wejście–wyjście. Funkcja $\hat{f}$ jest przybliżeniem prawdziwej (lecz nieznanej) funkcji gęstości f rozkładu prawdopodobieństwa przynależności punktów przestrzeni do klas w danym zadaniu. Zadanie klasyfikacji polega na odgadnięciu etykiety $l_j \in L$ zadanego „nowego" obiektu x_j. Ponieważ wejściowe obiekty mogą zajmować dowolne położenia w przestrzeni cech, zadanie klasyfikacji sprowadza się *de facto* do odgadnięcia (w postaci niejawnej) funkcji f.

W statystycznej teorii rozpoznawania obrazów zakłada się, iż łączny rozkład gęstości prawdopodobieństwa $p(x)$ wystąpienia wektora x jest superpozycją cząstkowych rozkładów prawdopodobieństwa (Duda i Hart, 1973):

$$p(x) = \sum_{j=1}^{c} p(\omega_j) p(x|\omega_j), \tag{1.2}$$

gdzie:

$0 \le p(\omega_j) \le 1$ – prawdopodobieństwo *a priori* klasy ω_j, $j = 1, \ldots, c$;

$p(x|\omega_j)$ – funkcja gęstości prawdopodobieństwa klasy ω_j, $j = 1, \ldots, c$.

Prawdopodobieństwo *a priori* $p(\omega_j)$ określa globalną proporcję obiektów klasy ω_j względem wszystkich obiektów generowanych przez źródło. Należy również wprowadzić pojęcie prawdopodobieństwa *a posteriori* $p(\omega_j|x)$, które określa szansę na to, iż konkretny obiekt *x* należy do klasy ω_j.

Reguła Bayesa określa prawdopodobieństwo *a posteriori* $p(\omega_j|x)$ formułą wiążącą ze sobą czynniki równania (1.2):

$$p(\omega_j|x) = \frac{p(\omega_j)p(x|\omega_j)}{p(x)};\ j = 1, \ldots, c. \tag{1.3}$$

Reguła decyzyjna:

$$K_B(x) = \omega_i \Leftrightarrow p(\omega_i|x) > p(\omega_j|x), \forall j \neq i, \tag{1.4}$$

z remisami rozstrzyganymi arbitralnie, nazywana jest klasyfikatorem Bayesa. Wprowadźmy funkcję kosztu $\lambda(\omega_i|\omega_j)$ (ang. *loss function*), określającą stratę wynikłą z przypisania obiektu klasy ω_j do klasy ω_i. Każdy klasyfikator minimalizujący zadaną funkcję kosztu nazywany jest klasyfikatorem optymalnym (Nilsson, 1965). Jeśli funkcja kosztu ma postać:

$$\lambda(\omega_i|\omega_j) = 1 - \delta_{ij}, \tag{1.5}$$

gdzie δ_{ij} jest symbolem Kroneckera, tj. przyjmuje wartość 1, gdy $i = j$, zaś 0 w pozostałych przypadkach, to taką funkcję nazywamy symetryczną funkcją kosztu. Nietrudno pokazać, iż przy założeniu symetrycznej funkcji kosztu, klasyfikator Bayesa jest klasyfikatorem optymalnym (Duda i Hart, 1973). W rozważaniach zamieszczonych w niniejszej rozprawie założono, że funkcja kosztu jest symetryczna.

Niestety, w praktyce rozkład prawdopodobieństwa przynależności obiektów do klas zwykle nie jest znany, a tym samym nie jest możliwe skonstruowanie klasyfikatora Bayesa. Co więcej, nie jest nawet zazwyczaj możliwe wiarygodne oszacowanie *błędu Bayesa*, tj. minimalnego błędu klasyfikacji w danym zadaniu, czyli ideału, do którego rzeczywiste klasyfikatory mogą się jedynie zbliżać.

Proces klasyfikacji składa się z kilku etapów. Przykładowo w zadaniach dotyczących diagnostyki lekarskiej mogą one wyglądać następująco:

a) decyzja o wyborze cech używanych do opisu stanu zdrowia pacjenta (czyli wybór badań, które należy przeprowadzić i uwzględnienie wybranych danych osobowych, np. wieku czy płci);
b) akwizycja danych (np. wprowadzenie wyników badań pacjenta do komputera, akwizycja obrazu z kamery);
c) obróbka wstępna (np. standaryzacja cech, odrzucenie lub uzupełnienie w zbiorze uczącym rekordów z brakującymi atrybutami);
d) ekstrakcja i/lub selekcja cech;
e) klasyfikacja przy użyciu wybranej metody.

Ekstrakcja cech jest pojęciem ogólniejszym od wyboru i pozyskania cech. Projektant systemu rozpoznawania obrazów może — korzystając z pomocy eksperta, np. lekarza — zamiast dwóch zmierzonych cech użyć jednej cechy sztucznej będącej np. ich iloczynem.

W praktyce nie zawsze występują wszystkie etapy od a) do e), nieraz też zmieniona jest ich kolejność. Przykładowo niektóre metody klasyfikacji — np. ważona wersja reguły najbliższego sąsiada z projekcją cech (*weighted Nearest Neighbor with Feature Projection*, wNNFP), p. Rozdz. 2.5 — są niewrażliwe na skalowanie cech oraz w łatwy, bezpośredni sposób radzą sobie z brakującymi atrybutami; w takich przypadkach z obróbki wstępnej można zrezygnować. Podobnie może być z selekcją cech. Etapy a) i b) mogą też występować w odwrotnej kolejności — przykładem będzie klasyfikacja poszczególnych piksli obrazu, gdzie akwizycja danych często poprzedza decyzję o wyborze cech, które w takim przypadku są zwykle „sztuczne” (gradient, wybrana miara lokalnej aktywności tekstury itp.).

1.1.2. Typy cech

Cechy (atrybuty), którymi opisane są obiekty, mogą być liczbowe (ang. *numeric*) bądź symboliczne (ang. *symbolic*, *nominal*). Cechy liczbowe można podzielić na ciągłe (ang. *continuous*) — np. poziom cukru we krwi, i dyskretne (ang. *discrete*) — np. liczba dzieci. Cechy symboliczne przyjmują wartości nieliczbowe, wśród których nie ma wprowadzonego porządku. Przykładem takiej cechy jest *kolor*, który może przyjąć jedną z np. sześciu wartości: *biały*, *zielony*, *czerwony*, *niebieski*, *żółty* lub *czarny*.

Szczególną klasą cech symbolicznych są cechy binarne, tj. o dziedzinie dwuwartościowej (cechą taką może być np. posiadanie/nieposiadanie prawa jazdy).

1.1.3. Miara odległości

Powszechnie przyjętym paradygmatem klasyfikacji (predykcji) jest podobieństwo. Można tę zasadę sformułować krótko: „jesteś taki, jak twoje otoczenie". Niektóre klasyfikatory korzystają z sąsiedztwa danej próbki w sposób bezpośredni (np. reguła *k* najbliższych sąsiadów), inne w sposób bardziej zakamuflowany (np. sieci neuronowe), ale nie sposób sobie wyobrazić odrzucenia tego paradygmatu. Aby jednak mówić o podobieństwie, należy wprowadzić pewną miarę podobieństwa czy też miarę odległości.

W przypadku cech liczbowych najpopularniejszymi miarami są metryka euklidesowa i miejska (Manhattan). Są one szczególnymi przypadkami rodziny metryk Minkowskiego, którą opisuje wzór:

$$d_p(x, y) = \left[\sum_{i=1}^{n} |x_i - y_i|^p \right]^{1/p}, \tag{1.6}$$

gdzie $d_p(x, y)$ jest odległością od wektora *x* do wektora *y*, *p* liczbą naturalną dodatnią specyfikującą konkretną miarę z rodziny Minkowskiego, natomiast *n* — wymiarem przestrzeni (często na oznaczenie wymiaru używa się też symbolu *d*). Dla $p = 1$ otrzymujemy metrykę miejską, zaś dla $p = 2$ euklidesową. Niekiedy używa się też miary maksimum, tj. przypadku granicznego przy $p \to \infty$.

Istnieją przesłanki sugerujące wyższość metryki miejskiej nad metryką euklidesową w zadaniach klasyfikacji oraz konstrukcji algorytmów szybkiego szukania najbliższego sąsiada. Dokładniej omówiono tę kwestię w Rozdz. 5.2.

Od lat znanych jest także wiele innych metryk, m. in. metryka Czebyszewa, Camberra, Mahalanobis i inne (Michalski i in., 1981). Ich praktyczna popularność jest jednak ograniczona, m. in. ze względu na często wyższy koszt liczenia takich odległości.

Obsługa cech symbolicznych jest z natury trudniejsza. Najprostsze, ale — jak łatwo odgadnąć — mało efektywne rozwiązanie to metryka dyskretna (ang. *overlap distance*):

$$d_i(x, y) = \begin{cases} 0; x_i = y_i \\ 1; x_i \neq y_i \end{cases}, \; i = 1, \ldots, d \tag{1.7}$$

Najpopularniejszą z niebanalnych metryk dla cech symbolicznych jest metryka dystrybucyjna *Value Difference Metric* (VDM) (Stanfill i Waltz, 1986), która doczekała się też kilku modyfikacji (Creecy i in., 1992; Cost i Salzberg, 1993). Uwzględnia ona częstości występowania poszczególnych wartości danych cech, zarówno w powiązaniu, jak i bez powiązania z etykietami klas obiektów, dla których te wartości cech wystąpiły. W mierze tej dwie wartości symboliczne uważane są za bliskie sobie (podobne), jeśli w podobny sposób skorelowane są z etykietą klasy. W pracy (Wilson i Martinez, 1997a) zaproponowano metrykę heterogeniczną HVDM (*Heterogeneous Value Difference Metric*), która używa znormalizowanej odległości VDM dla cech symbolicznych, znormalizowanej odległości euklidesowej dla cech liczbowych, zaś w przypadku, gdy przynajmniej dla jednego z rozpatrywanych obiektów nie jest określona wartość danej cechy, przyjmowana jest po tej cesze odległość 1.

Metryki VDM i HVDM są przykładami miar odległości, w których brane są pod uwagę estymowane rozkłady prawdopodobieństwa przynależności próbek do klas. Mówi się, iż są to metryki lokalne, tj. zwracające wartość uzależnioną od położenia punktów w przestrzeni (tj. wrażliwe np. na translację konkretnej pary punktów, dla których liczona jest odległość) — w odróżnieniu od metryk globalnych, np. Minkowskiego. Zainteresowanie metrykami lokalnymi od początku lat 80-tych jest duże (Short i Fukunaga, 1980; Short i Fukunaga, 1981; Aha i Goldstone, 1992; Friedman, 1994; Hastie i Tibshirani, 1995; Ricci i Avesani, 1995; Blanzieri i Ricci, 1999), jednak zaleta takiego podejścia, czyli dostosowanie się do konkretnego zadania, może być zarazem wadą: metryki lokalne mają więcej parametrów, a zatem są bardziej wrażliwe na szum (efekt przeuczenia; p. także Rozdz. 3.1). Spośród wymienionych propozycji bazujących na rozkładzie prawdopodobieństw *a posteriori*, dwie zasługują na szczególną uwagę ze względu na ich własności teoretyczne. Metryka Shorta i Fukunagi (1981) minimalizuje wartość oczekiwaną różnicy między błędem dla skończonego zbioru a (estymowanym) błędem asymptotycznym, zaś metryka minimalnego ryzyka (*Minimal Risk Metric*) Blanzieriego i Ricciego (1999) bezpośrednio minimalizuje błąd dla danego (skończonego) zbioru.

1.1.4. Brakujące wartości atrybutów

Bolączką projektantów schematu klasyfikacji bywają niekompletne dane. Sytuacja taka ma miejsce, gdy w części obiektów (ze zbioru uczącego i/lub klasyfikowanego) brakuje jednej lub wielu wartości cech. Przyczyną tego stanu rzeczy może być uszkodzenie

mechaniczne danych (np. zatarcie pisma ręcznego na karcie choroby pacjenta), niedbałość operatora, awaria sprzętu, gromadzenie danych z różnych i nie do końca kompatybilnych źródeł *etc*.

Najczęściej w praktyce rekordy z brakującymi danymi dla wygody usuwa się (można to uczynić, gdy takich przypadków jest bardzo mało; w przeciwnym razie strata informacji może być zbyt duża), przyjmuje znormalizowaną odległość 0 lub 1 po tej cesze (możliwość przyjęcia jednego z dwóch skrajnych wariantów dobitnie świadczy, jak słabo rozumiany jest ten problem) lub zastępuje brakującą wartość cechy średnią lub medianą tej cechy po całym zbiorze.[2] Rozważano jednak subtelniejsze podejścia. W pracy (Lee i in., 1976) brakująca wartość cechy zastępowana jest średnią wartością tejże cechy po kilku najbliższych sąsiadach rozpatrywanej próbki. Dixon (1979) porównuje sześć metod obsługi problemu i konkluduje, iż najlepszą jest metoda zakładająca, że odległość do brakującej cechy jest taka sama, jak średnia odległość po wszystkich parach wektorów rzutowanych na tę jedną cechę. W przypadku, gdy ubytków w zbiorze jest stosunkowo niewiele, ww. metoda Lee wydaje się porównywalnie dobra. Jeszcze bardziej złożoną — i dającą dobre wyniki — metodę zaproponowano w kontekście drzew decyzyjnych (Quinlan, 1993). Popularny schemat C4.5 estymuje prawdopodobieństwa przyjmowania przez brakującą cechę poszczególnych wartości i uwzględnia wszystkie owe potencjalne wartości stosując klasyfikację równolegle i podejmując finalną decyzję w wyniku głosowania z wagami.

1.1.5. Ocena jakości klasyfikatora

Jakość (niezbyt szczęśliwe tłumaczenie angielskiego terminu *accuracy*) klasyfikacji, tj. trafność predykcji klasyfikatora w danym zadaniu, estymowana jest wzorem:

$$acc := 1 - r/n\,, \tag{1.8}$$

gdzie r jest liczbą próbek mylnie klasyfikowanych, zaś n liczbą wszystkich poddawanych klasyfikacji próbek. Iloraz r/n jest estymowanym *błędem* klasyfikacji (ang. *classification error*).

Niestety, wzór (1.8) nie wyjaśnia, jaki zbiór/zbiory ma(-ją) być podstawą estymacji jakości klasyfikatora. Problem polega na ustaleniu zbioru uczącego i zbioru testowego (lub wielu par takich zbiorów) dla danego zbioru odniesienia (zadania). Jak

[2] Efektowny przykład wskazujący na słabość tejże metody, niezależnie od jej odmiany, podaje Jankowski (1999, str. 140): „gdy założyć, że wartości brakujące zastąpi się najczęściej występującą wartością danej cechy, można spodziewać się, że brakująca informacja o wzroście niemowlęcia wyniesie ok. 165 cm".

podzielić dostępny zbiór na część uczącą (czyli służącą do konstrukcji klasyfikatora) i część testową (czyli służącą do oceny jakości danego klasyfikatora)? Innymi słowy, jak dokonać partycji dostępnego zbioru? Jeśli zbiór uczący będzie zbyt mały, to niska będzie jakość jego predykcji oraz duża wariancja wyników przy różnych losowych partycjach. Z drugiej strony, mały zbiór testowy powoduje niepewną estymację błędu. W literaturze brakuje silniejszych stwierdzeń nt. wyższości którejś z metod podziału nad innymi. Dodatkowo w literaturze panuje pewne zamieszanie terminologiczne dotyczące metod estymacji błędu. W praktyce, stosuje się zazwyczaj jedną z czterech poniższych metod estymacji (terminologia zaczerpnięta z monografii (Kurzyński, 1997) oraz pracy przekrojowej (Jain i in., 2000)):

- metoda resubstytucji (ang. *resubstitution method*) — cały dany zbiór jest używany zarówno do uczenia, jak i do testowania. Metoda ta nie jest polecana, gdyż daje zawyżone (zbyt optymistyczne wyniki), zwłaszcza przy małych (w stosunku do wymiarowości) zbiorach;
- metoda wydzielania (ang. *holdout method*) — losowa połowa zbioru służy do konstrukcji klasyfikatora, a druga połowa do jego testowania. Wadą metody jest pesymistyczna estymacja jakości klasyfikatora;
- metoda minus jednego elementu (ang. *leave-one-out method*), zwana też metodą usuwania — klasyfikator generowany jest n–1 razy, tj. dla każdego (n–1)-elementowego podzbioru pełnego zbioru, podczas gdy zbiorem testowym dla każdego wygenerowanego klasyfikatora jest tylko jedna pozostała próbka. Estymacja błędu jest w tej metodzie nieobciążona (tj. sprawiedliwa), ale wariancja błędu jest znaczna; ponadto nie jest łatwo osiągnąć satysfakcjonującą szybkość działania tej metody. Należy zauważyć, że również w metodzie resubstytucji klasyfikowane są pojedyncze próbki z użyciem pozostałej części pełnego zbioru, ale różnica polega na tym, że w metodzie resubstytucji uczenie przeprowadzane jest raz (dla całego zbioru), zaś w metodzie minus jednego elementu aż n razy;
- metoda k-krotnej walidacji skrośnej, zwana również metodą rotacji (ang. *k-fold cross validation*, *rotation method*), stanowi kompromis pomiędzy metodą wydzielania a metodą minus jednego elementu: dostępny zbiór dzielony jest losowo na k możliwie równych podzbiorów (ang. *folds, plies* (Skalak, 1997)), a następnie każda z k części jest po kolei zbiorem testowym, zaś pozostałe k–1 części zbiorem uczącym. Oznacza to, iż przy estymacji błędu należy

wygenerować k klasyfikatorów (modeli). Błąd estymacji tej metody jest stosunkowo niski (generalnie tym niższy, im większe k), wariancja błędu jest niższa niż przy metodzie minus jednego elementu, zaś koszt czasowy realizacji dla typowych w praktyce wartości k (5–10) umiarkowany. Metoda ta jest obecnie najczęściej stosowana w praktyce.

Gwoli uniknięcia nieporozumień, należy podkreślić, że opisane metody dotyczą estymacji błędu w fazie klasyfikacji. Jednakże metoda minus jednego elementu jest również stosowana przy wyborze parametrów klasyfikatora na etapie uczenia. Faza uczenia ma za zadanie dobrać jeden zestaw parametrów klasyfikatora, a zatem metoda minus jednego elementu w tej fazie oznacza — w przeciwieństwie do podanej wyżej definicji — iż dla rozpatrywanego zestawu parametrów, tj. dla konkretnego modelu estymowany jest błąd przy n-krotnym użyciu podzbiorów (n–1)-elementowych klasyfikujących pozostałą próbkę. Należy zwrócić uwagę, że przy metodzie minus jednego elementu użytej do testowania klasyfikatora znajdowanych jest aż n modeli.

W większości testów przeprowadzanych w tej pracy została zastosowana metoda 5-krotnej walidacji skrośnej, przy czym procedurę tę powtarzano pięciokrotnie (tj. dla pięciu losowych partycji zbioru na pięć równych części) w celu zwiększenia statystycznej wiarygodności wyników.

1.2. Główne typy klasyfikatorów

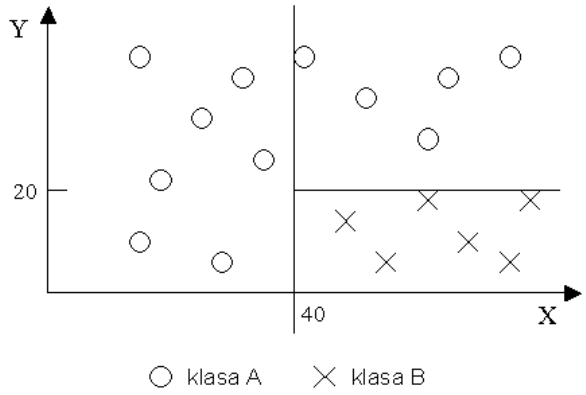
Na przestrzeni ostatnich kilkudziesięciu lat opracowano niezliczone algorytmy klasyfikacji, różniące się walorami użytkowymi (jakość klasyfikacji, szybkość klasyfikacji, szybkość uczenia, zapotrzebowania pamięciowe *etc*.), złożonością i stopniem podatności na analizy teoretyczne. Większość algorytmów klasyfikacji można jednak zaliczyć do jednej z kilku głównych rodzin metod rozpoznawania. Ze świadomością, iż zestawienie nie jest kompletne, autor rozprawy chciałby poniżej przedstawić kilka głównych typów klasyfikatorów: drzewa decyzyjne, sieci neuronowe, klasyfikatory minimalnoodległościowe oraz rodzinę naiwnych klasyfikatorów Bayesa.

1.2.1. Drzewa decyzyjne

Drzewa decyzyjne (ang. *decision trees*) działają w oparciu o zasadę „dziel i rządź". Polega ona, w jednej z odmian, na podziale złożonego problemu na prostsze,

a następnie rekursywnym zastosowaniu tej strategii do stworzonych podzadań. Najpopularniejszymi algorytmami opracowanymi na bazie drzew decyzyjnych są „dychotomizer interaktywny" ID3 (*Interactive Dichotomizer*, *version 3*) (Quinlan,1979), CART (Breiman i in., 1984; skrót nazwy pochodzi od tytułu oryginalnej pracy „Classification and Regression Trees") oraz C4.5 (Quinlan, 1993).

Drzewo decyzyjne jest skierowanym acyklicznym grafem, w którym każdy wierzchołek jest albo wierzchołkiem decyzyjnym posiadającym dwóch lub więcej potomków, albo liściem. Z liściem skojarzona jest klasa. Klasyfikowana próbka jest niejako poddawana serii pytań, przechodząc po każdej odpowiedzi w dół drzewa, począwszy od korzenia, a skończywszy na liściu, w którym otrzymuje ona etykietę klasy. Drzewo decyzyjne może być więc opisane w postaci reguł. Dla przykładu z Rys. 1.1, drzewo decyzyjne może działać według schematu opisanego regułami na Rys. 1.2, przy założeniu, że klasa „kółek" oznaczona będzie przez A, zaś klasa „krzyżyków" przez B.



Rys. 1.1. Drzewo decyzyjne dla przykładowego zadania dwudecyzyjnego

```
IF (X<40) THEN output(class A)
        ELSE
IF (Y<20) THEN output(class B)
        ELSE output(class A)
```

Rys. 1.2. Zestaw reguł równoważny drzewu decyzyjnemu z Rys. 1.1

Podstawowym problemem przy konstrukcji drzewa decyzyjnego jest kryterium rozszczepienia wierzchołka (ang. *splitting rule*). Zbudowanie minimalnego, w sensie liczby wierzchołków, drzewa decyzyjnego poprawnie klasyfikującego wszystkie obiekty zbioru uczącego, jest problemem NP-zupełnym (Hyafil i Rivest, 1976). Zazwyczaj do rozszczepienia wierzchołka używa się heurystyki „spoglądającej" na jeden krok

naprzód. Zasadniczo chodzi o to, aby szybko dojść do wierzchołków zawierających elementy wyłącznie z jednej klasy, a zatem nie wymagających dzielenia (czyli będących liśćmi). Okazuje się jednak, że nasuwająca się naturalna heurystyka zachłanna, dzieląca wierzchołki tak, aby podzbiory wychodzące z danego wierzchołka były maksymalnie zróżnicowane, tj. zawierały możliwie różne proporcje klas, w praktyce nie działa dobrze (Breiman i in., 1984). Szerszą dyskusję problemu wyboru kryterium rozszczepiania zawierają m. in. prace (Martin, 1997; Gama, 1999).

Jeśli powstałe drzewo decyzyjne jest proste, to jest to fakt wielostronnie korzystny: duża jest szybkość klasyfikacji, małe wymagania pamięciowe, a także — co w niektórych zastosowaniach może być najważniejsze — wygenerowane reguły są czytelne dla człowieka. Niestety, w praktyce często mamy do czynienia z danymi zaszumionymi, które komplikują drzewo decyzyjne, utrudniają zrozumienie jego reguł, a także narażają na przeuczenie (p. Rozdz. 3.1). Próbą pokonania tych problemów jest obcięcie (ang. *pruning*) drzewa. Za cenę utraty zgodności (ang. *consistency*), tj. poprawnego rozpoznania absolutnie wszystkich obiektów zbioru uczącego, mamy nadzieję poprawić jakość klasyfikacji nieznanych próbek. Redukcja polega na zastąpieniu odpowiednio głęboko położonych wierzchołków liśćmi. Wyróżnia się dwie grupy metod: metody zatrzymujące w odpowiednim momencie generację drzewa (ang. *pre-pruning*) oraz metody generujące pełne drzewo, a następnie dokonującego jego „odchudzenia" (ang. *post-pruning*). Bardziej obiecujące, choć wolniejsze, są metody z drugiej grupy (Breiman i in., 1984; Quinlan, 1993).

Podsumowując: zaletami drzew decyzyjnych są odporność na skalowanie i inne ściśle monotoniczne transformacje cech, mała wrażliwość na pojedyncze próbki „odstające" od reszty (ang. *outliers*), elastyczność (poszczególne poddrzewa nie muszą być utworzone wg tych samych modeli), względna odporność na zbędne cechy, względna zrozumiałość wyników (wygenerowanych reguł) dla człowieka (ang. *comprehensibility*) i szybkość klasyfikacji.

Do wad podejścia należy za to zaliczyć trudność w aproksymacji prostych, lecz nierównolegle do osi ułożonych granic decyzyjnych,[3] niestabilność algorytmu (małe zmiany na wejściu powodują duże zmiany w strukturze drzewa) oraz replikację (ang. *replication*), tj. tendencję do powielania sekwencji pytań w osobnych poddrzewach, co

[3] Drzewa dokonujące rozszczepienia w oparciu o wiele cech (ang. *multivariate trees*) pokonują ten problem, ale za cenę wydłużonego czasu uczenia i mniejszej zrozumiałości wynikowego zbioru reguł dla człowieka.

powoduje spadek jakości predykcji i nieoszczędną, trudną dla zrozumienia, reprezentację. Podział danych na mniejsze podzbiory sprawia, iż nie można wziąć pod uwagę więcej niż log *n* cech, przy założeniu równomiernego rozszczepiania zbioru — co może być niewystarczające, jeśli liczba znaczących cech jest duża. Dodatkowo obsługa brakujących wartości cech jest problematyczna i nastręczająca programistycznych trudności. Cytat: „Około połowy kodu schematu CART i 80 procent wysiłku programistycznego pochłonęła obsługa brakujących wartości!" (Friedman i in., 1996).

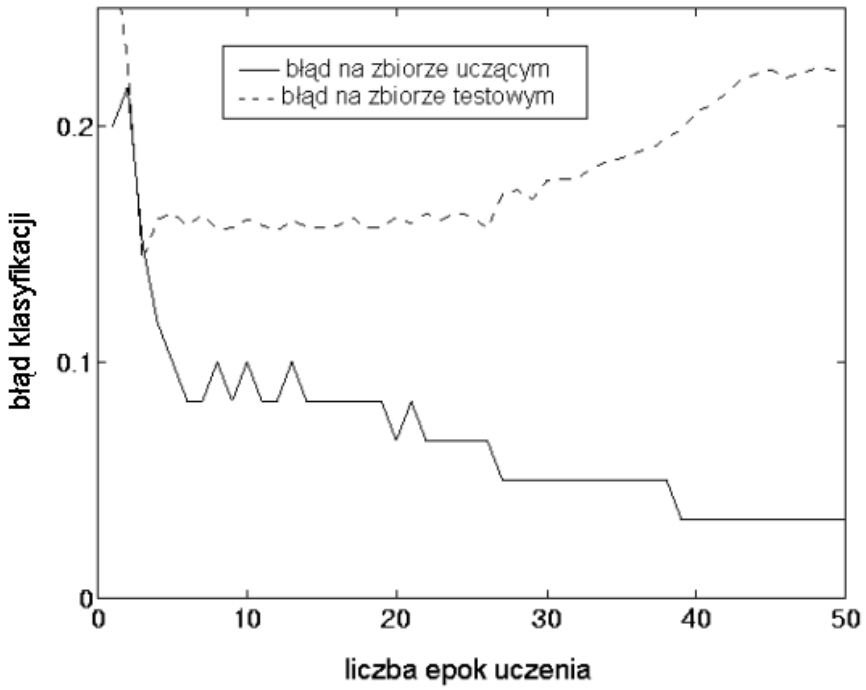
1.2.2. Sieci neuronowe

Sztuczna sieć neuronowa (ang. *artificial neural network*, ANN) jest układem komunikujących się jednostek („neuronów", węzłów), działającym równolegle. Neurony ułożone są w warstwach. Z każdym połączeniem między neuronami skojarzona jest waga, która wpływa na modyfikację (najczęściej nieliniową) przekazywanych dalej przez neuron wartości. Proces uczenia sieci w zadaniu klasyfikacji nadzorowanej polega — w najprostszym ujęciu — na modyfikacji wag celem możliwie dobrej dyskryminacji próbek wejściowych pochodzących z różnych klas.

Rodzina sieci neuronowych jest zbyt bogata, aby autor rozprawy mógł dokonać tu choćby najogólniejszej taksonomii. Wspomnijmy — w porządku historycznym — o takich koncepcjach, jak model neuronu McCullocha i Pittsa (1943), uczenie neuronów w sieci (Hebb, 1949), sieci liniowe (perceptron (Rosenblatt, 1958; Minsky i Papert, 1969), sieć ADALINE (Widrow i Hoff, 1960)), perceptrony wielowarstowe, w tym z propagacją wsteczną (Rumelhart i in., 1986), sieci wykorzystujące kwantyzację wektorową (*Learning Vector Quantization*, LVQ) (Kohonen, 1986), sieć przesyłająca żetony (*Counterpropagation*) (Hecht-Nielsen, 1987), sieci o radialnych funkcjach bazowych (*Radial Basis Functions Networks*, RBF *Networks*) (Broomhead i Lowe, 1988; Moody i Darken, 1989; Chen i in., 1991) oraz sieci probabilistyczne (*Probabilistic Neural Networks*, PNN) (Specht, 1990; 1992). W latach 90-tych wiele nowatorskich prac dotyczyło konstrukcji zespołu sieci neuronowych, podejmującego decyzje np. w wyniku głosowania większościowego. Koncepcje te zostaną omówione szerzej w Rozdz. 3.

Większość sieci neuronowych uczy się z wykorzystaniem kryterium błędu średniokwadratowego (ang. *mean square error*, MSE). Takie kryterium zakłada, iż etykiety klas próbek w zbiorze są zaszumione szumem gaussowskim rozłożonym symetrycznie wokół środka klastra. Założenie takie często sprawdza się przy aproksymacji funkcji dowolnego sygnału, ale zawodzi przy klasyfikacji, tj. przy dyskretnych

wyjściach (Rimer i in., 2002). Co się z tym wiąże, w literaturze zgłaszano wiele przypadków eksperymentów, w których nauczona sieć oferowała niski błąd na zbiorze uczącym, lecz dużo większy na zbiorze testowym (Andersen i Martinez, 1999; Schiffmann i in., 1993). Opisany efekt przeuczenia (ang. *overfitting*) powstaje także wtedy (jak sugeruje polska nazwa), gdy sieć poddawana była procesowi uczenia zbyt długo — istotnie, dobór kryterium końca procesu uczenia jest jednym z podstawowych problemów rozpoznawania obrazów, a szczególnie ważnym w dziedzinie sieci neuronowych. Przykładowy efekt przeuczenia na 60-elementowym zbiorze uczącym, osiągnięty przez sieć neuronową zawierającą 10 neuronów ukrytych, obrazuje Rys. 1.3, zaczerpnięty z pracy (Jain i in., 2000). Przez wszystkie epoki uczenia (50) błąd na zbiorze uczącym niemal monotonicznie maleje, natomiast najmniejszy błąd na zbiorze testowym osiągnięty byłby, gdyby uczenia zaprzestano już po kilku epokach.



Rys. 1.3. (Jain i in., 2000) Przedstawienie rozbieżności między błędem estymowanym w czasie uczenia (linia ciągła) a błędem na zbiorze testowym (linia przerywana). Oś rzędnych określa błąd klasyfikacji, oś odciętych — czas (epokę uczenia).

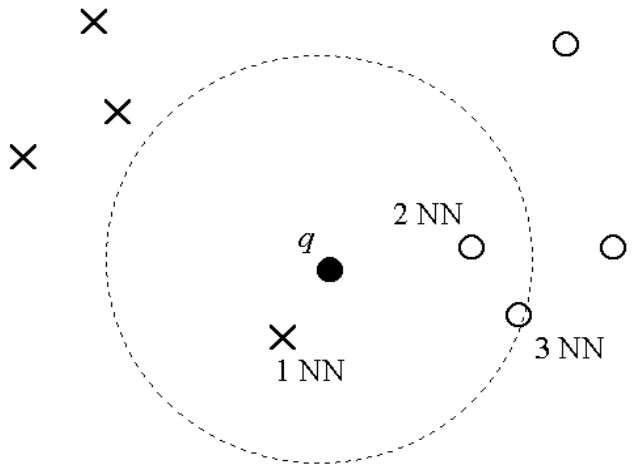
Zainteresowanie sieciami neuronowymi jest nadal duże i w ostatnich latach włożono wiele wysiłku w ograniczanie wad tego podejścia. Przykładowo, techniki takie jak *Quickprop* (Fahlman, 1988) i *Rprop* (Riedmiller i Braun, 1993) mają na celu przyspieszenie uczenia sieci z propagacją wsteczną poprzez dynamiczną zmianę

parametrów uczenia. Zhang (1994) zmniejszył zbiór uczący ograniczając go jedynie do próbek najbardziej reprezentatywnych i powiększając go tylko wtedy, gdy wydaje się to potrzebne do prawidłowej zbieżności procesu uczenia. Modnym kierunkiem badań w ostatnich latach jest użycie algorytmów ewolucyjnych do wyboru architektury pojedynczej sieci (Branke, 1995) lub całego zespołu (Yao i Liu, 1998; Friedrich, 1998).

W Polsce ukazało się kilka monografii poświęconych sieciom neuronowym, m. in. prace Tadeusiewicza (1993) i Rutkowskiej i in. (1997).

1.2.3. Klasyfikatory minimalnoodległościowe

Najpopularniejszym klasyfikatorem minimalnoodległościowym jest reguła *k* najbliższych sąsiadów (ang. *k Nearest Neighbors*, *k*-NN), przypisująca nieznaną próbkę do klasy najczęściej występującej wśród jej *k* najbliższych, w sensie ustalonej miary odległości, sąsiadów (Fix i Hodges, 1951; Fix i Hodges, 1952). Remisy rozstrzygane są arbitralnie (warto zauważyć, że dla częstego w praktyce przypadku dwóch klas, dowolne nieparzyste *k* eliminuje remisy). Działanie reguły *k*-NN dla $k = 3$ jest ukazane na Rys. 1.4. Ponieważ wśród trzech najbliższych sąsiadów próbki testowej *q* są dwa „kółka" i jeden „krzyżyk", próbka *q* zostanie przypisana do klasy „kółek". Podkreślić należy, że kolejność *k* najbliższych sąsiadów (w sensie ich odległości od próbki testowej) nie ma wpływu na wynik klasyfikacji.



Rys. 1.4. Reguła 3-NN. Próbka testowa *q* zostanie przypisana do klasy „kółek".

Zalety *k*-NN:

- asymptotyczna optymalność, tj. zbieżność błędu do błędu Bayesa, gdy liczność zbioru $n \to \infty$ i $k/n \to 0$ (Fix i Hodges, 1951);
- w praktyce na ogół wysoka jakość klasyfikacji;
- stosunkowo szybkie uczenie (wybór *k*) i selekcja cech;
- prostota koncepcji/implementacji i łatwość wprowadzania modyfikacji;
- możliwość estymacji błędu na etapie uczenia przy pomocy metody minus jednego elementu.

Wady *k*-NN:

- wolna klasyfikacja;
- konieczność przechowywania całego zbioru odniesienia w pamięci;
- duża, w porównaniu z wieloma innych klasyfikatorami, wrażliwość na zbędne cechy.

Z pewnego punktu widzenia także stabilność *k*-NN, tj. małe różnice w predykcji przy dodaniu/usunięciu niewielu próbek ze zbioru odniesienia, jest wadą, a mianowicie wynika z niej trudność w zaprojektowaniu zespołu możliwie różniących się (ang. *diverse*) klasyfikatorów. Szerzej kwestia ta zostanie omówiona w Rozdz. 3.

Najszybszą i najprostszą odmianą *k*-NN jest reguła 1-NN, przypisująca nieznaną próbkę do klasy jej jednego najbliższego sąsiada.

Zalety 1-NN:

- zazwyczaj dość wysoka jakość klasyfikacji — w projekcie Statlog (Michie i in., 1994), w ramach którego porównano ok. 20 algorytmów klasyfikacji różnych typów na ponad 20 dużych rzeczywistych zbiorach danych, aż dla 75% testowanych zbiorów najlepszą wartością *k* dla *k*-NN była właśnie 1 (Feng i in., 1993), ale doświadczenia autora rozprawy nie potwierdzają tej obserwacji;
- brak uczenia, o ile pominie się selekcję cech;
- skrajna prostota;
- możliwość redukcji zbioru odniesienia w celu przyspieszenia klasyfikacji;
- możliwość aproksymacji bardziej złożonego klasyfikatora, jeśli zbiór odniesienia zostanie zreklasyfikowany klasyfikatorem aproksymowanym (np. *k*-NN).

Wady 1-NN:

- nadal wolna klasyfikacja (choć oczywiście nieco szybsza niż przy *k*-NN; w odróżnieniu od *k*-NN także nie występuje problem z szybkością w najgorszym przypadku);
- znaczna wrażliwość na szum;
- brak asymptotycznej optymalności (Cover i Hart, 1967);
- konieczność przechowywania całego zbioru odniesienia w pamięci;
- duża wrażliwość na zbędne cechy;
- pewne trudności w obsłudze cech symbolicznych, zwłaszcza przy niektórych algorytmach redukcji.

Najważniejszym wynikiem teoretycznym dotyczącym reguły 1-NN jest oszacowanie asymptotycznego błędu parą nierówności:

$$p^* \le p(e) \le p^*\left(2 - \frac{c}{c-1}\,p^*\right), \tag{1.9}$$

gdzie p^* jest błędem Bayesa, $p(e)$ błędem reguły 1-NN, zaś c — liczbą klas (Cover i Hart, 1967). Ponadto autorzy wykazali, iż ograniczeń (1.9) nie można poprawić w ogólnym przypadku.

Reguła *k*-NN doczekała się szeregu modyfikacji (Dasarathy, 1991). Jedną z najwcześniejszych była jej ważona odmiana (ang. *weighted k*-NN), w której waga sąsiada zadanej próbki q uzależniona była od jego odległości od q (Dudani, 1976). Recepcja tej idei była zróżnicowana: krytyczna w (Bailey i Jain, 1978) oraz aprobująca w (Morin i Raeside, 1981; Zavrel, 1997). Ważonej *k*-NN użyto także w systemie FIBL (*Fuzzy Instance-Based Learning*) z automatycznym doborem funkcji opadania wag (Wilson, 1997). W eksperymentach autora rozprawy ważona *k*-NN była nieznacznie gorsza od oryginalnej. Mimo pewnej przewagi w niektórych testach literaturowych nad zwykłą *k*-NN, wydaje się, że główną zaletą wersji ważonej jest jej większa odporność na zmiany wartości parametru *k*.

Tomek (1976a) zaproponował wprowadzenie progu *k'* oznaczającego minimalną liczbę sąsiadów z danej klasy potrzebną do przypisania danej próbki do tej klasy. Koncepcja ta daje możliwość odrzucania niepewnych wyników klasyfikacji (np. jeśli tylko trzech z pięciu sąsiadów jest z klasy pierwszej, zaś dwóch z drugiej).

Rozmyta reguła *k*-NN (ang. *fuzzy k*-NN) (Jóźwik, 1983; Keller i in., 1985) poszerza przestrzeń poszukiwań poprzez zastąpienie „twardych" etykiet (ang. *hard labels, crisp labels*) próbek zbioru uczącego etykietami rozmytymi o stopniach przynależności do poszczególnych klas, które w pewnym sensie oddają charakter sąsiedztwa danej próbki. Algorytm Jóźwika (1983) podaje procedurę uczenia etykiet rozmytych i w przeciwieństwie do schematu Kellera nie bazuje na odległościach. Idea Jóźwika została pozytywnie oceniona w pracy (Bezdek i in., 1986).

Wettschereck i Dietterich (1994) zaproponowali cztery wersje reguły *k*-NN z lokalnym, tj. osobnym dla każdej zadanej próbki, wyborem *k*. Choć autorzy twierdzą, że w sytuacji dużego zaszumienia lub zmieniającej się lokalnie istotności cech w zadaniu lokalne warianty *k*-NN mogą znacząco dominować nad wersją oryginalną, to jednak wyniki zaproponowanych metod na zbiorach UCI (Merz i Murphy, 1996) są średnio bardzo zbliżone do wyników standardowej *k*-NN.

Interesującą koncepcję, pod nazwą „*k* dyplomatycznych najbliższych sąsiadów" (*k Diplomatic Nearest Neighbors*, *k*-DNN), przedstawili Sierra i in. (2000). Reguła ta szuka *k* sąsiadów z każdej klasy osobno, a następnie wybiera tę klasę, dla której średnia odległość opisanych sąsiadów do testowej próbki jest najmniejsza. Jakość klasyfikacji oferowana przez tę regułę nierzadko przewyższa jakość oryginalnej *k*-NN, a dodatkową jej zaletą jest małe prawdopodobieństwo remisu przy dokonywaniu decyzji.

W Rozdz. 7 wprowadzono schemat reguły *k*-NN z głosowaniem po wielu wartościach parametru *k*. Schemat ten w naturalny sposób można zastosować i do innych klasyfikatorów stosujących predykcję na bazie *k* wyselekcjonowanych sąsiadów (np. *k*-NCN).

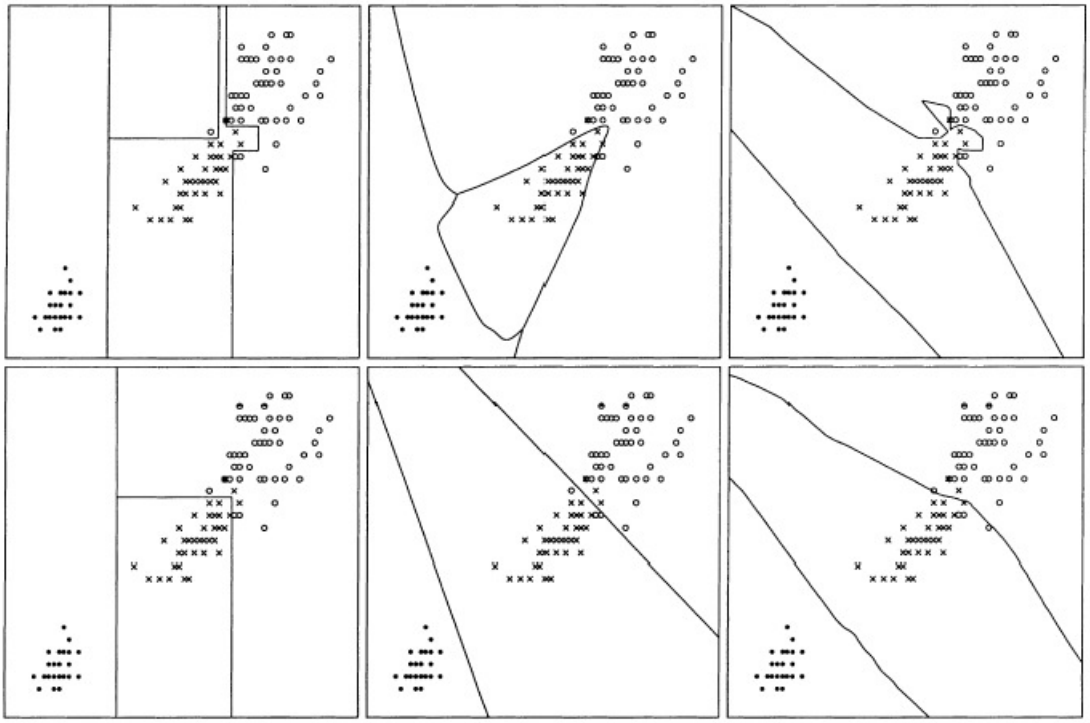
Rys. 1.5 zaczerpnięty został z pracy (Andersson i in., 1999). Zostały na nim — na przykładzie zbioru Iris (Fisher, 1936) rzutowanego na dwa wymiary — ukazane efekty działania klasyfikatorów trzech typów: drzew decyzyjnych ID3 (kolumna lewa), sieci neuronowych z propagacją wsteczną (kolumna środkowa) i reguły *k*-NN (kolumna prawa). Dolny rząd rysunków obejmuje klasyfikatory o prostszych (bardziej gładkich) granicach decyzyjnych w stosunku do odpowiednich klasyfikatorów rzędu górnego.

Dokładniej, zaprezentowano granice decyzyjne:

- pełnego drzewa i drzewa po zastosowaniu obcięcia gałęzi (*pruning*);
- sieci neuronowej z 30 węzłami ukrytymi uzyskanej po 26500 epok uczenia oraz sieci neuronowej z jedynie dwoma węzłami ukrytymi po 20000 epok uczenia;

- reguły 1-NN i 10-NN.

Zbiór Iris zawiera 150 obiektów z trzech klas. Jak widać z rysunku, mniejsza złożoność klasyfikatora — prostsza budowa drzewa decyzyjnego czy mniejsza liczba węzłów ukrytych w sieci neuronowej — może prowadzić do bardziej efektywnej generalizacji. Odwrotne zjawisko zachodzi często dla klasyfikatorów typu najbliższy sąsiad: większa liczba sąsiadów zazwyczaj pociąga za sobą większe wygładzenie granic decyzyjnych, co jest na ogół korzystne.



Rys. 1.5. (Andersson i in., 1999) Porównanie granic decyzyjnych indukowanych przez klasyfikatory różnych typów i różnej złożoności na zbiorze Iris

1.2.4. Naiwny klasyfikator Bayesa

Tzw. naiwny klasyfikator Bayesa (ang. *Naive Bayes Classifier,* NBC) to interesująca koncepcja teoretyczna, zakładająca niezależność cech w zadaniu (Duda i Hart, 1973). Gdyby cechy były niezależne, wówczas prawdopodobieństwo warunkowe przynależności próbki x do danej klasy ω_i byłoby iloczynem prawdopodobieństw przynależności x przy rozkładach rzutowanych na poszczególne cechy; formalnie:

$$p(x|\omega_i) = p(x_1|\omega_i) \cdot \ldots \cdot p(x_d|\omega_i), \tag{1.10}$$

gdzie x_1, ..., x_d są wartościami poszczególnych cech próbki x. Jeśli założenie niezależności cech jest spełnione, wówczas naiwny klasyfikator Bayesa jest klasyfikatorem

asymptotycznie optymalnym. Ponieważ estymacja $p(x_j|\omega_i)$, $j=1, ..., d$, jest łatwa (i szybka), więc opisany fakt można wykorzystać do konstrukcji klasyfikatora. W praktyce NBC jest klasyfikatorem mało przewidywalnym: na zbiorach wspomnianego wcześniej projektu Statlog spisywał się generalnie słabo, ponosząc spektakularną klęskę na zbiorze Letters złożonym z 26 klas i dotyczącym rozpoznawania pisma: 52,9% błędów, podczas gdy błąd reguły *k*-NN wyniósł 6,8% (Michie i in., 1994), natomiast wg pracy (Domingos i Pazzani, 1996) NBC przewyższył klasyczny schemat drzewa decyzyjnego C4.5 na 16 z 28 zbiorów UCI. W wymienionej pracy Domingos i Pazzani potwierdzili wcześniejsze spostrzeżenie (m. in. (Clark i Niblett, 1989; Dougherty i in., 1995)), iż NBC osiąga nieraz bardzo dobrą jakość, mimo dużej korelacji między cechami. Wytłumaczono to zjawisko poprzez wskazanie na zasadniczą różnicę między klasyfikacją a estymacją prawdopodobieństwa: klasyfikator może zwracać poprawne predykcje, nawet gdy estymacje prawdopodobieństwa przynależności próbki do poszczególnych klas są obarczone dużym błędem. Co więcej, autorzy podali kilka warunków koniecznych i dostatecznych dla optymalności klasyfikatora NBC. Szybkość uczenia i klasyfikacji tego algorytmu zachęcają do dalszych badań nad zrozumieniem jego właściwości i możliwością wykrycia zastosowań, w których będzie on osiągał zadawalającą jakość.

Warto także wspomnieć o kilku modyfikacjach NBC określanych wspólną nazwą *semi-naive Bayes classifiers* („do pewnego stopnia naiwne klasyfikatory Bayesa"), polegających na selekcji (Langley i Sage, 1994) i łączeniu (ang. *joining*) cech (Kononenko, 1991; Pazzani, 1996), wyborze podzbioru zbioru uczącego dla klasyfikacji danej próbki (Langley, 1993) czy modyfikacji prawdopodobieństw zwracanych przez standardowy NBC (Webb i Pazzani, 1998). Generalnie jednak warianty te nie wnoszą dużej poprawy jakości.

Bardziej pomyślne były próby wykorzystania klasyfikatora NBC lokalnie (drzewo *NBTree* (Kohavi, 1996), w którym decyzję w każdym liściu podejmuje osobno nauczony naiwny klasyfikator Bayesa) oraz w zespole klasyfikatorów używających różnych zestawów cech (Zheng, 1998). Do najlepszych jakościowo algorytmów tego kierunku należy naiwny klasyfikator Bayesa z doborem sąsiedztwa (*Selective Neighborhood Naive Bayes*, SNNB) (Xie i in., 2002), wybierający lokalnie jeden z wygenerowanych *on-line* (tj. dla zadanej próbki) modeli, których zróżnicowanie wynika z użycia różnych podzbiorów pełnego zbioru odniesienia. Niestety, ceną takiego podejścia jest liniowy

w liczności zbioru czas klasyfikacji, co negatywnie kontrastuje z „typowym" naiwnym klasyfikatorem Bayesa.

1.3. Porównywanie klasyfikatorów

Empiryczne porównywanie klasyfikatorów jest zadaniem niezwykle trudnym. W literaturze spotyka się niezliczone, często sprzeczne, stwierdzenia nt. wyższości jednego algorytmu/podejścia nad innym. Tymczasem nawet kryteria obiektywnej oceny algorytmów nie zawsze są jasne (lub autorzy przemilczają niekorzystne aspekty własnego algorytmu, dotyczące np. szybkości uczenia lub szybkości klasyfikacji). Subiektywne kryteria, takie jak łatwość implementacji algorytmu, są także w praktyce bardzo ważne, a nie wiadomo czy i jak podobne aspekty można mierzyć.

Trudności w porównywaniu algorytmów wynikają m. in. z takich czynników jak:

- trudność z doborem dużych reprezentatywnych dostępnych publicznie zbiorów danych;
- wrażliwość, niekiedy bardzo duża, wielu algorytmów lub zbiorów od takich czynników, jak dobór metryki, rodzaj selekcji cech lub jej brak, standaryzacja danych lub jej brak, sposób doboru parametrów klasyfikatora;
- brak lub nieadekwatność statystycznej oceny wiarygodności wyników;
- różne metodologie testów (walidacja skrośna z różnymi parametrami k, metoda resubstytucji i inne).

Niektóre z wymienionych problemów można uznać za obiektywne (np. duża wrażliwość klasyfikatora na dobór parametrów), inne spowodowane są błędami metodologicznymi spotykanymi w literaturze.

Interesujące zestawienie na temat metodologii przeprowadzanych eksperymentów opublikował Prechelt (1996). Prześledził on 190 artykułów poświęconych sieciom neuronowym (pomijając prace czysto teoretyczne lub dotyczące np. realizacji sprzętowych sieci), m. in. z takich pism jak „Neural Networks" i „IEEE Transactions on Neural Networks", z lat 1993–94 i stwierdził m. in., iż tylko 8% prac zaprezentowało wyniki na więcej niż jednym zbiorze rzeczywistym, w 29% prac nie został użyty ani jeden zbiór rzeczywisty czy tzw. realistyczny (nietrywialny zbiór sztuczny), zaś jedna trzecia artykułów nie przedstawia jakościowego porównania metody autorskiej z jakimkolwiek innym klasyfikatorem. Jeszcze inny problem polega na tym, że zbiory, na których eksperymentowano, są często małe, co zapewne wiąże się z długotrwałym

procesem uczenia sieci neuronowych. Należy jednak przyznać, że od tamtego czasu średni poziom prac zawierających porównania klasyfikatorów, jak się wydaje, wzrósł.

Rozpoznawanie obrazów jest dziedziną dynamicznie rozwijającą się od kilkudziesięciu lat. Należy w tym miejscu wspomnieć o monografiach i pracach przekrojowych ułatwiających orientację w przedmiocie i rozlicznych jego aspektach. Do najważniejszych monografii poświęconych rozpoznawaniu obrazów należą prace Dudy i Harta (1973), Fukunagi (1990), Dasarathy'iego (ed., 1991) i Mitchella (1997). Z nowszych prac przekrojowych podejmujących większość głównych zagadnień dziedziny, należy wyróżnić obszerny artykuł Jaina i in. (2000). W Polsce monografie przedmiotu wydali m. in. Jajuga (1990), Tadeusiewicz i Flasiński (1991) oraz Kurzyński (1997). Ponadto ukazało się kilka monografii autorów polskich poświęconych wybranym rodzinom klasyfikatorów, m. in. praca Tadeusiewicza (1993) o sieciach neuronowych i Cichosza (2000) o uczeniu na bazie drzew decyzyjnych i systemów reguł.

Rozdział II
Selekcja cech

Selekcja cech (atrybutów) jest jednym z najważniejszych zagadnień w takich dziedzinach, jak rozpoznawanie obrazów (*pattern recognition*), uczenie maszynowe (*machine learning*), wyszukiwanie informacji w wielkich bazach danych (*data mining*) oraz pokrewnych. Odrzucenie zbędnych cech prowadzi do zmniejszenia wymagań pamięciowych, większej efektywności działania algorytmów szybkiego szukania bliskich sąsiadów, a więc i szybszego wyszukiwania przybliżonego w bazach danych, zmniejszenia złożoności czasowej algorytmów uczenia oraz — co zwykle najważniejsze — polepszenia wyników klasyfikacji (zdolności generalizacji danego klasyfikatora). Selekcja cech, poprzez wskazanie najistotniejszych atrybutów zbioru uczącego, prowadzi do „skupienia się" algorytmu uczenia na tych aspektach danych, które są najbardziej użyteczne.

W rozdziale przedstawiono trudności związane z rozważanym zagadnieniem, opisano techniki pokrewne, takie jak ważenie i standaryzacja cech, oraz przedstawiono bliżej kilka algorytmów klasycznej selekcji cech. Opisano także klasyfikator „najbliższy sąsiad z projekcją cech" (*Nearest Neighbor with Feature Projection*, NNFP) (Akkus i Güvenir, 1996), będący *de facto*, zwłaszcza w wersji zmodyfikowanej (Kubat i Chen, 1998), niekonwencjonalnym podejściem do problemu selekcji cech, a także podano autorską metodę optymalnej redukcji zbiorów dla tego klasyfikatora. Ponadto zaproponowano ideę szybkiej implementacji strategii przyrostowego dodawania (*Forward Selection Strategy*, FSS) i usuwania (*Backward Selection Strategy*, BSS) cech (Kittler, 1978), jak również ideę implementacji pełnego przeglądu zestawów cech z wykorzystaniem kodu Gray'a (Grabowski, 2000b). Strategie FSS, BSS oraz ich modyfikacje zostały zweryfikowane empirycznie.

2.1. Specyfika zagadnienia i problemy pokrewne

Jeśli w zbiorze odniesienia niektóre cechy są zbędne, to działają jak szum, pogarszając jakość klasyfikacji. W przypadku asymptotycznym (zbiór o nieskończonej liczności) skończona liczba nawet czysto losowych cech nie może pogorszyć jakości. Dla

skończonych zbiorów uczących pokazano jednak (Langley i Iba, 1993), że liczba próbek w zbiorze musi rosnąć w tempie wykładniczym, aby zachować jakość klasyfikacji w przypadku, gdy dodawane są zbędne cechy.

Można wyróżnić dwa typy zbędnych cech: nieistotne (ang. *irrelevant*) i nadmiarowe (ang. *redundant*). Cechy nieistotne to cechy nieskorelowane z etykietami klas. Cechy nadmiarowe zaś, to cechy, których wartości można wyliczyć z wartości pozostałych cech. Dla przykładu, jeśli pewna cecha jest dokładnym powtórzeniem innej, tj. dla każdego elementu zbioru wartości tych dwu cech są jednakowe, to oczywiście jedną z tych cech można uznać za nadmiarową. Jest to najprostszy przypadek nadmiarowości, ale nie jedyny. Przykładowo dana cecha może być iloczynem dwóch innych cech.

Selekcja cech polega tradycyjnie na wybraniu możliwie dobrego podzbioru cech z pełnego zestawu wejściowego. Przez „dobry" podzbiór cech można rozumieć zestaw nie zawierający opisanych wyżej cech zbędnych. Warto wspomnieć, że istnieją także (nieliczne) podejścia, w których na wyjściu nie otrzymujemy podzbioru cech (Domingos, 1997; Bay, 1998). Najczęstszym schematem jest sekwencyjny przegląd zestawów cech według pewnej strategii (np. dodając po jednej — losowej lub w szczególny sposób wybranej — cesze) i ocena jakości każdego zestawu.

Rozważane zagadnienie ma długą historię i było przedmiotem licznych badań; do klasycznych prac można zaliczyć m. in. (Fu, 1968) i (Mucciardi i Gose, 1971). Spośród bardziej oryginalnych koncepcji należy wymienić takie podejścia do selekcji cech, jak użycie algorytmów genetycznych (Brill i in., 1992; Vafaie i De Jong, 1992 i in.), teorii informacji (Koller i Sahami, 1996; Singh i Provan, 1996), wykorzystanie wymiaru fraktalnego (Traina i in., 2000) czy łączenie selekcji cech i redukcji zbioru odniesienia (Skalak, 1994). Rozważano także selekcję cech symbolicznych (Baim, 1988; Rauber i Steiger-Garção, 1993).

Alternatywną dla selekcji cech fazą uczenia jest ważenie cech (ang. *feature weighting*). W najprostszej wersji polega ono na przypisaniu każdej cesze wartości z przedziału [0, 1], będącej mnożnikiem danej cechy przy liczeniu odległości (Kelly i Davis, 1991; Salzberg, 1991; Lee, 1994; Mohri i Tanaka, 1994; Lowe, 1995). Należy zauważyć, że jeśli dopuszczamy tylko wagi 0 i 1, to proces ten sprowadza się do klasycznej selekcji cech. Istnieją także bardziej złożone schematy, w których wagi przypisywane są z uwzględnieniem kontekstu. I tak postulowano m. in. wprowadzenie osobnej wagi dla każdej występującej w zbiorze odniesienia wartości każdej z cech

(Stanfill i Waltz, 1986; Nosofsky i in., 1989) lub osobnej wagi dla każdej cechy w obrębie każdej klasy (Aha, 1989). Najbardziej złożone algorytmy używają osobnych wag dla każdej cechy każdej próbki zbioru, więc można je uznać za w pełni kontekstowe. W tej klasie algorytmów wagi dobierane są na etapie uczenia (Aha i Goldstone, 1992) lub klasyfikacji (Atkeson i in., 1997; Hastie i Tibshirani, 1996). Ważenie cech znacząco poszerza przestrzeń poszukiwań, więc potencjalnie umożliwia lepsze przetworzenie zbioru niż klasyczna selekcja cech. Niestety, w praktyce ryzyko przeuczenia parametrów jest duże. Jest to zjawisko stale występujące w zagadnieniach przetwarzania obrazów. Podobne założenia (i podobne niebezpieczeństwa) kryją się w eksperymentach z funkcją odległości dobieraną do danego zadania.

Prostą i często stosowaną formą ważenia cech jest standaryzacja (normalizacja) (ang. *standardization*) cech. Warto zauważyć, że np. w przypadku, gdy dwie cechy przyjmujące podobne wielkości fizyczne wyrażone są w zupełnie różnych jednostkach (np. jedna w centymetrach, a druga w metrach), klasyfikator będzie niejako faworyzował cechę o większym rozrzucie nominalnych wartości, a zatem wynik finalny będzie wypaczony. Standaryzacja polega na transformacji wszystkich cech do podobnie szerokich przedziałów przyjmowanych wartości. Najczęściej spotyka się trzy odmiany standaryzacji (Jajuga, 1990):

- klasyczną (po transformacji liniowej średnia wartość każdej cechy wynosi 0, a odchylenie standardowe 1);
- medianową (analogiczna do poprzedniej, lecz wykorzystująca medianę i odchylenie medianowe);
- skalującą liniowo do przedziału [0, 1].

Wadą dwu z przedstawionych metod w oryginalnej postaci, za wyjątkiem standaryzacji medianowej, jest wrażliwość na próbki znacznie różniące się (na jednej lub wielu cechach) od pozostałych (ang. *outliers*). „Nietypowe" próbki są często wynikiem błędów pozyskania danych. Dla cechy, na której pojawił się „odszczepieniec", efektywny przedział dla typowych wartości będzie miał szerokość dużo mniejszą niż 1, a to oznacza, iż cecha ta będzie miała mniejszy wpływ na klasyfikację od innych. Aby pokonać ten problem, Skalak (1997) w wariancie skalującym do przedziału [0, 1] nadał nową wartość 0 (1) wszystkim wartościom mniejszym (większym) od średniej dla danej cechy o przynajmniej 3-krotną wielokrotność odchylenia standardowego dla tej cechy. Podobne rozwiązanie użyte zostało w schemacie ReMind (Cognitive Systems, Inc., 1992).

W praktyce wszystkie opisane metody standaryzacji działają zazwyczaj porównywalnie dobrze; mimo iż wartości cech rzeczywistych zbiorów bardzo często nie przyjmują rozkładu normalnego, to jednak rzadkie są sytuacje, w których np. standaryzacja klasyczna (wykorzystująca odchylenia standardowe wartości przyjmowanych przez cechy) prowadzi do znacznie zwiększonych błędów klasyfikacji (Aha, 1990).

Jeszcze inną techniką pokrewną selekcji cech jest wprowadzanie sztucznych cech będących kombinacjami cech danych (Matheus i Rendell, 1989; Wnek i Michalski, 1994). W ten sposób utworzone nowe atrybuty — np. różnica pewnej pary cech traktowana jako nowa cecha — mogą spowodować znaczne przyspieszenie zbieżności algorytmu klasyfikacji i/lub poprawić jego szybkość po dalszej selekcji. Przykładem niech będzie iloczyn cech 3 i 4 zbioru Iris (Fisher, 1936), uwzględniany jako jedyna cecha dla klasyfikatora. Jakość klasyfikacji jest wówczas porównywalna z jakością przy pełnym zbiorze (Gama, 1999), zaś szybkość klasyfikacji np. przy regule 1-NN wzrasta ok. 4-krotnie. Rzecz jasna, w przypadku ogólnym wybór operatora i cech, na których należy go zastosować, nie jest zadaniem łatwym.

2.2. Modele oceny zestawów

Wyróżnia się dwa schematy oceny zestawów cech: metody, w których jakość przeglądanych zestawów estymuje się przy użyciu tej samej reguły decyzyjnej, która będzie użyta w następnym etapie do klasyfikacji (model „*wrapper*"), oraz metody, gdzie przeglądane zestawy są oceniane przy pomocy innego kryterium (model „*filter*"). Innymi słowy, w modelu „filter" abstrahujemy od klasyfikatora używanego w danym zadaniu. Pierwsza klasa algorytmów, jak można oczekiwać, daje na ogół lepsze wyniki (John i in., 1994; Aha i Bankert, 1995), kosztem jest jednak dłuższy proces działania algorytmu. W modelu „filter" kryterium oceny jest na ogół znacznie szybsze niż ocena jakości klasyfikatora, zatem i cała sesja selekcji cech jest stosunkowo szybka — jednak inny sposób oceny zestawów cech na etapie uczenia niż na etapie klasyfikacji niemal zawsze prowadzi do znalezienia gorszego zestawu. Do najbardziej znanych algorytmów typu „filter" należą RELIEF (Kira i Rendell, 1992) i FOCUS (Almuallim i Dietterich, 1991), gdzie klasyfikatorami używanymi w zadaniu są drzewa decyzyjne. W pracy (Cardie, 1993) użyto jednak filtracji dla klasyfikatora 1-NN.

W pozostałej części rozdziału rozważa się podejścia typu „wrapper".

2.3. Pełny przegląd zestawów cech

Strategie selekcji cech na ogół w pewnym porządku przeglądają wybrane zestawy, estymują ich jakość (jak wyżej wspomniano, metodą typu „wrapper" bądź „filter") i zwracają ten zestaw, dla którego estymowany błąd był najmniejszy. Nasuwa się pytanie: czy nie można przejrzeć wszystkich zestawów cech (oczywiście w takim przypadku kolejność przeglądania nie ma znaczenia)? Niestety, w większości przypadków jest to niemożliwe ze względu na czas selekcji rosnący wykładniczo z wymiarem. Przy d cechach wejściowych istnieje $2^d - 1$ niepustych kombinacji cech, a zatem w praktyce pełny przegląd wykonalny jest jedynie wtedy, gdy d nie przekracza ok. 12–15. Jednakże w przypadkach, w których pełny przegląd jest możliwy, jest on zalecany (o ile $n >> d$ (gdzie n jest licznością zbioru), gdyż w przeciwnym razie pełny przegląd zestawów cech może doprowadzić do przeuczenia).

Ponieważ w dalszej części pracy wielokrotnie pojawiają się złożoności przedstawianych algorytmów, warto przypomnieć, iż symbol $O(f(\cdot))$ oznacza, że koszt algorytmu, wyrażany zwykle liczbą elementarnych operacji, jest asymptotycznie ściśle ograniczony z góry iloczynem $c \cdot f(\cdot)$, gdzie c jest pewną stałą, zaś $f(\cdot)$ funkcją jednego lub więcej parametrów związanych z zadaniem (np. liczność zbioru, wymiar) lub specyfiką algorytmu (np. zadaną liczbą sąsiadów) (Aho i in., 2003). Niejednokrotnie złożoność algorytmu w najgorszym przypadku różni się od złożoności algorytmu w przypadku średnim, i za wyjątkiem oczywistych sytuacji warto podawać obie złożoności.

Rozważmy zatem pełny przegląd zestawów cech dla klasyfikatora typu 1-NN przy zastosowaniu modelu „wrapper". Załóżmy, że do oceny jakości danego zestawu użyta zostanie metoda minus jednego elementu. Ponieważ znalezienie najbliższego sąsiada wymaga policzenia wszystkich odległości do innych próbek w zbiorze (jeśli nie są stosowane struktury indeksujące służące do szybkiego znajdowania najbliższego sąsiada; efektywność tych metod w wysokich wymiarach jest jednak problematyczna — p. Rozdz. 4 i 5), więc koszt oceny pojedynczego zestawu wynosi $O(d \cdot n^2)$. Uwzględniając liczbę wszystkich zestawów cech, wnioskujemy, że koszt pełnego przeglądu zestawów cech wynosi $O(2^d \cdot d \cdot n^2)$.

W pracy (Grabowski, 2000b) autor rozprawy przedstawił ideę redukującą ten koszt do $O(2^d \cdot n^2)$. Jej istotą jest korzystanie przy testowaniu danego zestawu cech

z informacji uzyskanych wcześniej. Pomysł wykorzystuje kod Gray'a, tj. sposób kodowania binarnego elementów zbioru wyliczeniowego (np. kolejnych liczb naturalnych od 1 do *n*), przy którym kody dwóch kolejnych elementów różnią się tylko na jednej pozycji (bicie).

W Tab. 2.1 przedstawiono przykładową formułę pozwalającą otrzymywać kody Gray'a dla liczb naturalnych; przyjęto arbitralnie, iż będą to liczby zapisywane na 8 bitach, tj. z przedziału [0, 255], gdzie najmłodszy bit w zapisie położony jest najbardziej na prawo. Operator *xor* to alternatywa wyłączająca, zaś *shr* to przesunięcie bitowe o jedną pozycję w prawo (tj. podzielenie przez 2 z utratą najmłodszego bitu).

Tabela 2.1: Przykładowy kod Gray'a
dla kilku najmniejszych liczb naturalnych

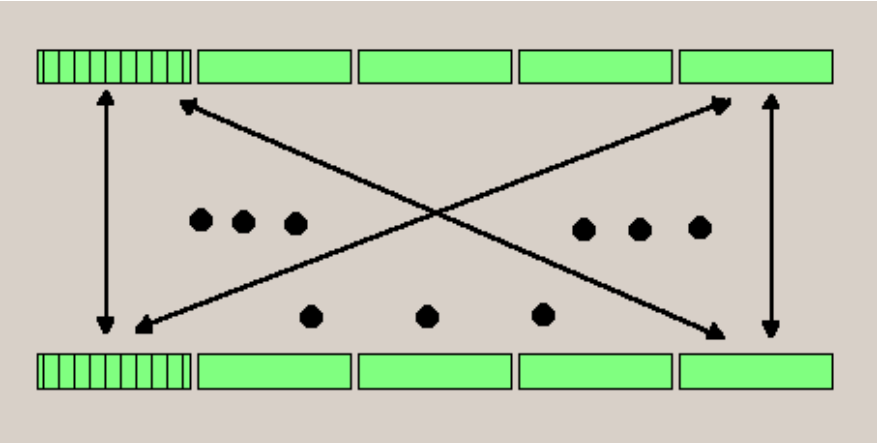
i	Gray(i) := i xor (i shr 1)
0	00000000
1	00000001
2	00000011
3	00000010
4	00000110
5	00000111
6	00000101

Dany zestaw cech może być opisany sekwencją *d* bitów, gdzie jedynka na *i*-tej pozycji oznacza, że *i*-ta cecha została włączona do zestawu, zaś zero, że tej cechy w zestawie nie ma. Używając *d*-pozycyjnego kodowania Gray'a można zatem ułożyć zestawy cech w taki sposób, aby dwa kolejne różniły się tylko co do jednej cechy. Jeżeli została zapamiętana odległość między dwoma obiektami przy poprzednim zestawie cech, to obliczenie odległości dla nowego zestawu wymaga jedynie dodania lub odjęcia odległości po nowej cesze (jest tak nie tylko w metryce miejskiej, ale w każdej metryce Minkowskiego, jeśli dla przyspieszenia pomija się liczenie pierwiastka).

Implementacja, w której zapamiętywane byłyby wszystkie odległości między elementami zbioru uczącego, byłaby jednak często pamięciowo zbyt kosztowna. Na szczęście można kolejno rozpatrywać pojedyncze próbki. Dla każdej z nich należy przejść wszystkie zestawy cech, zapamiętując na pojedynczych bitach informację, czy przy danym zestawie bieżąca próbka jest dobrze klasyfikowana (przyjmujemy, że jedynka oznacza poprawną klasyfikację). Koszt pamięciowy wynosi teraz $O\left(2^d \cdot n\right)$. Typowo przy tej strategii 2^d bitów to co najwyżej 1–2 KB (tj. gdy *d* nie przekracza ok. 13–14), a zatem implementacja ma zastosowanie dla zbiorów o wielkości do ok. 100

tysięcy próbek (przy jeszcze większych zbiorach można zastosować kilka przejść — w każdym przejściu analizowanych jest tylko część zestawów cech). Jeżeli obliczone dane pamiętane są w macierzy, w której dany wiersz informuje, jak dana próbka jest klasyfikowana przez poszczególne zestawy, to ostateczny wynik — wskazanie najlepszego globalnie zestawu — uzyskiwany jest przez znalezienie kolumny zawierającej największą liczbę jedynek.

Przy implementacji — niekoniecznie na potrzeby selekcji cech — procedury estymacji błędu (przy użyciu zbioru testowego lub metodą minus jednego elementu dla zbioru uczącego), warto pamiętać o wykorzystaniu pamięci podręcznej (*cache*). Przykładowo dla metody minus jednego elementu, odległości między próbkami warto liczyć w małych grupach (Grabowski, 2000b). Dokładniej ideę tę wyjaśnia Rys. 2.1. Grupy próbek (na rysunku jest ich pięć) są na tyle małe, że każda para grup (jest ich 10 w prezentowanym przykładzie) mieści się w całości w pamięci podręcznej. Przy założeniu estymacji błędu dla reguły 1-NN, globalne minimum odległości dla danej próbki q znajdowane jest wśród minimów odległości do próbek w poszczególnych grupach zawierających próbkę q. Innymi słowy, po pierwszym kroku opisanej idei dla próbki q, znani są jej najbliżsi sąsiedzi w poszczególnych grupach. Wykonanie kilku dodatkowych porównań odległości prowadzi do znalezienia najbliższego sąsiada q w całym zbiorze. Analogiczną ideę można zrealizować również przy estymacji błędu dla reguły k-NN.



Rys. 2.1. Idea wykorzystania pamięci *cache* przy estymacji błędu na zbiorze uczącym regułą minus jednego elementu. Odległości dla par próbek liczone są w obrębie kolejnych par grup

W teście na pełnym zbiorze RB8.PAT liczącym 4297 próbek o 64 cechach i należących do 14 klas, opisana idea przyspieszyła estymację błędu metodą minus jedne-

go elementu dla klasyfikatora 1-NN blisko dwukrotnie (z ok. 73 s do 39 s na komputerze Celeron 466). Rzecz jasna, przyspieszenie możliwe jest jedynie o czynnik stały, jednak, jak się wydaje, w najbliższych latach dysproporcja między szybkością pamięci konwencjonalnych a pamięci *cache* pogłębi się na niekorzyść pamięci konwencjonalnych.

Kolejną, bardzo oczywistą, optymalizacją jest przerywanie estymacji błędu dla danego zestawu, gdy liczba źle klasyfikowanych próbek przekracza już aktualne minimum i tym samym dany zestaw nie ma szans na zwycięstwo. I tym razem redukcja czasu była znacząca: w testach autora przeważnie od 20% do 50%; w skrajnych przypadkach znacznie więcej.

Należy wspomnieć, iż w niektórych zastosowaniach celem selekcji cech nie jest wybranie zestawu cech o najlepszej predykcji ze wszystkich podzbiorów pełnego zbioru d cech, lecz wybranie najlepszego podzbioru k cech, gdzie k jest zadane z góry. W takim przypadku, jeśli d i k są dostatecznie małe, alternatywą dla przeglądu $\binom{d}{k}$ k-elementowych podzbiorów cech jest algorytm ograniczający (*branch-and-bound*) (Narendra i Fukunaga, 1977), który gwarantuje znalezienie najlepszego k-elementowego zestawu. Niestety, opiera się on na dość rzadko spełnianym w praktyce założeniu monotoniczności jakości podzbiorów cech (tj. zakłada, iż dla dowolnego m-elementowego zbioru cech, jakość jego dowolnego (m–1)-elementowego podzbioru będzie nie większa od jakości zbioru m-elementowego). Co więcej, w najgorszym przypadku procedura Narendry i Fukunagi może wykonywać się nawet dłużej od pełnego przeglądu. Istnieją także szybsze wersje algorytmu *branch-and-bound* (Yu i Yuan, 1993; Somol i in., 2000), szczególnie korzystne, gdy zadana liczba cech jest mała w stosunku do oryginalnej (Kudo i Sklansky, 2000), ale złożoność najgorszego przypadku pozostaje niezmieniona.

2.4. Strategie dołączania i odłączania cech

W strategii FSS (*Forward Selection Strategy*), tj. kolejnego dołączania cech (Kittler, 1978), wybiera się najpierw cechę, która użyta samodzielnie oferuje najmniejszy błąd klasyfikacji estymowany na zbiorze uczącym. Do tej cechy próbuje się dołączyć drugą z pozostałych d–1 cech, tak aby ponownie zminimalizować estymowany błąd. Proces kontynuowany jest aż do osiągnięcia pełnego zestawu d cech. Ostateczną odpowiedzią

jest zestaw najlepszy ze wszystkich rozpatrywanych. Jeśli pojawią się zestawy równoważne, to preferuje się ten, w którym ostatnio dołączona cecha, traktowana samodzielnie, oferuje mniejszy błąd klasyfikacji.

Analogicznie działa strategia BSS (*Backward Selection Strategy*), czyli kolejnego odrzucania cech (Marill i Green, 1962). Tym razem procedura zaczyna od pełnego zbioru d cech i kolejno odrzuca po jednej cesze, tak aby pomniejszony zestaw był w danym kroku najlepszy z możliwych. Spośród przejrzanych zestawów wybierany jest zestaw, z którym związany jest najmniejszy błąd klasyfikacji. Analogicznie do FSS, w sytuacji pojawienia się zestawów równorzędnych, preferowany jest ten, w którym ostatnio odrzucona cecha byłaby samodzielnie „gorsza".

Warto nadmienić, iż obie opisane strategie występują też w literaturze pod alternatywnymi określeniami: odpowiednio *Sequential Forward Selection* (SFS) i *Sequential Backward Selection* (SBS).

Obie strategie wymagają przeglądu $O\left(d^2\right)$ zestawów, jednak BSS jest wolniejsza, gdyż częściej analizuje zestawy złożone z większości cech wejściowych. Doak (1992) twierdzi, iż BSS jest lepszą strategią niż FSS, przeczą temu jednak wyniki z pracy (Aha i Bankert, 1995) oraz własne autora rozprawy (p. Rozdz. 3.5). Ogólnie mówiąc, słabość BSS wydaje się polegać na dość przypadkowym wyborze odrzucanych cech na początku swojego działania: w obecności wielu cech nietrudno wtedy o odrzucenie cechy dobrej. Szersze rozważania nt. porównania FSS i BSS zostaną przedstawione w Rozdz. 3.5, gdzie obie strategie przegrywają jednak (w sensie jakości klasyfikacji) z eksperymentalnymi wariantami klasyfikatora MFS (*Multiple Feature Subsets*) (Bay, 1998), używającego wielu zestawów cech do klasyfikacji. W cytowanej pracy Aha i Bankert (1995) potwierdzają także eksperymentalnie intuicję, iż BSS jest lepsza pod względem jakości od FSS, gdy liczba istotnych cech w zadaniu jest duża, natomiast przy małej liczbie istotnych cech lepsza jest FSS.

Podobnie jak przy pełnym przeglądzie, i tu można przyspieszyć ocenę danego zestawu regułą minus jednego elementu dla klasyfikatora 1-NN, korzystając z zestawu poprzedniego, zawierającego tylko o jedną cechę mniej lub więcej (Grabowski, 2000b). Załóżmy, że stosowana jest strategia FSS i w danym momencie zakończono testowanie zestawów złożonych z j cech. Znaleziony zatem został pewien lokalnie najlepszy zestaw j cech. W następnym kroku należy sprawdzić wszystkie $d - j$ zestawów powiększonych o jedną cechę. W proponowanej, szybkiej implementacji dla każdej

próbki po kolei liczone są i zapamiętywane odległości do wszystkich pozostałych próbek przy zestawie *j* cech. Następnie w drugiej tablicy odległości te są uaktualniane poprzez dodanie do wartości bieżących odległości do danej próbki obliczonej na pojedynczej testowanej cesze. Po policzeniu wszystkich odległości dla danego zestawu należy sprawdzić, jak dana próbka jest klasyfikowana i zapamiętać to na jednym bicie (załóżmy, że 1 oznacza poprawną klasyfikację). Proces powtarzany jest dla każdej z $d - j$ „brakujących" cech. Tym samym dla danej próbki otrzymujemy $d - j$ bitów informujących, jak została ona sklasyfikowana przez każdy z rozpatrywanych zestawów. Po wyznaczeniu wektorów binarnych klasyfikacji dla wszystkich próbek pozostaje znalezienie zestawu poprawnie klasyfikującego największą liczbę próbek. Jeżeli opisane wektory binarne kolejnych próbek będą wierszami w macierzy, to zadanie polega na znalezieniu kolumny zawierającej największą liczbę jedynek. Cecha reprezentująca tę kolumnę jest lokalnie najlepsza i zostanie przyłączona do zestawu. Analogiczne postępowanie możliwe jest przy strategii BSS. Należy zauważyć, że koszt liczenia pojedynczej odległości został zredukowany z $O(d)$ do $O(1)$.

Naturalnym rozszerzeniem strategii FSS i BSS jest „dwukierunkowość": w każdym kroku wolno albo dodać, albo usunąć jedną cechę, w zależności od tego, co daje lepszy wynik. Proces powtarzany jest aż do „zapętlenia" (np. na przemian wybierane są zestawy (1, 3, 4) i (1, 3)) lub gdy liczba iteracji przekroczy zadany z góry limit. Metodę tę nazwano w pracy roboczo „BD-SS" (*Bidirectional Selection Strategy*). Postanowiono zaczynać od pustego zbioru cech. Tym razem w przypadku remisów (tj. zestawów równoważnych) o wyborze cechy (dodanej lub usuniętej) decyduje ich kolejność.

Druga metoda najpierw wykonuje np. procedurę FSS, a potem startując ze znalezionego zestawu, procedurę BSS.[4] Operacje te są powtarzane, aż w danej sesji FSS lub BSS nie znajdzie już lepszego zestawu. Rzecz jasna, wynik ostateczny na zbiorze uczącym jest zawsze co najmniej równie dobry, jak wyłoniony pojedynczą strategią, która rozpoczyna selekcję. Ceną tego ulepszenia jest czas. Na potrzeby testów metodę nazwano „FSS+BSS" (do pierwszego etapu wybrano sesję FSS).

W pierwszym eksperymencie z opisanymi powyżej algorytmami selekcji cech przeprowadzonym przez autora rozprawy (Grabowski, 2000b) użyto zbioru FerritesOld złożonego z 5 klas, 12 cech i wylosowanych 1000 próbek (spośród 2885 próbek w

[4] A. Jóźwik, korespondencja prywatna, 2000.

całym zbiorze). Jeden to jeden z trzech wykorzystywanych w tej pracy zbiorów dotyczących kontroli jakości rdzeni magnetycznych, które były produkowane w zakładach „Polfer" w Warszawie. Liczba cech jest na tyle mała, że można dokonać również pełnego przeglądu cech. Pozwoli to w pewien sposób ocenić pozostałe strategie. Estymacja błędu została dokonana bez użycia zbioru testowego, tj. podawane wyniki dotyczą najlepszego z rozważanych zestawów cech ocenianych metodą minus jednego elementu. Używanym klasyfikatorem była reguła 1-NN. Tab. 2.2 przedstawia wyniki dla poszczególnych par klas.

Tabela 2.2: Wyniki strategii FSS, BSS, BD-SS i FSS+BSS oraz pełnego przeglądu na zbiorach liczących po 400 próbek, 12 cech i 2 klasy. We wszystkich przypadkach klasy równoliczne.

para klas	pełny przegląd		FSS		BSS		BD-SS		FSS+BSS	
	liczba cech	błąd [%]	liczba cech	błąd [%]	liczba cech	błąd [%]	liczba cech	błąd [%]	liczba cech	błąd [%]
1 / 2	2	0,00	10	0,00	11	0,00	1	0,25	10	0,00
1 / 3	5	3,75	5	4,00	5	3,75	5	4,00	5	4,00
1 / 4	6	4,00	7	4,75	7	4,75	6	4,00	7	4,75
1 / 5	3	0,25	3	0,25	10	0,25	4	0,25	3	0,25
2 / 3	1	0,00	1	0,00	11	0,25	1	0,00	1	0,00
2 / 4	1	0,00	1	0,00	11	0,25	1	0,00	1	0,00
2 / 5	5	2,50	7	3,00	10	2,75	2	3,50	7	3,00
3 / 4	6	24,00	6	24,50	10	24,25	7	24,25	6	24,50
3 / 5	3	0,50	3	0,50	3	0,50	3	0,50	3	0,50
4 / 5	2	0,25	2	0,25	4	0,50	2	0,25	2	0,25

Jak widać, trudno wskazać jednoznacznego zwycięzcę. Strategia FSS+BSS zwróciła zawsze identyczne wyniki jak FSS. Dla par 2 / 5 i 3 / 4 żaden niepełny przegląd nie znalazł optymalnego zestawu. Problem ze strategią BD-SS polega na tym, że nie może ona „wyjść" z minimum lokalnego. Pozytywną stroną tego zjawiska jest jednak przeważnie najkrótszy czas procesu selekcji wśród testowanych metod.

Do drugiego testu wykorzystano zbiór RB8.PAT (4297 obiektów, 64 cechy, 14 klas), również dotyczący kontroli jakości rdzeni magnetycznych. Większość par klas w tym zbiorze jest liniowo rozdzielna, często nawet bez selekcji cech, dlatego też do eksperymentów wybrano trzy „trudniejsze" przypadki. Przetestowano strategie: FSS, BSS, BD-SS, FSS+BSS oraz zmierzono jakość przy pełnym zestawie cech (tj. bez selekcji). I w tym przypadku nie używano osobnego zbioru testowego.

Tym razem (Tab. 2.3) najlepszą strategią dla wybranych zbiorów okazała się FSS+BSS. Strategia BSS ma tendencję do wybierania zestawów zawierających

większą liczbę cech niż w zestawach wybieranych przez pozostałe algorytmy. Średnio najmniej liczne zestawy generowała strategia BD-SS.

Tabela 2.3: Wyniki metod FSS, BSS, BD-SS i FSS+BSS na wybranych parach klas zbioru RB8.PAT; l.p. – liczba próbek, l.c. – liczba cech

para klas	l.p.	błąd [%]	FSS		BSS		BD-SS		FSS+BSS	
			l.c.	błąd [%]	l.c.	błąd [%]	l.c.	błąd [%]	l.c.	błąd [%]
4 / 13	315	1,27	5	0,32	7	0,32	5	0,00	5	0,32
7 / 8	215	10,23	10	1,86	20	5,58	3	2,79	10	1,86
4 / 8	352	3,69	22	0,57	42	0,28	9	1,14	21	0,28

2.5. Przegląd innych wybranych algorytmów selekcji cech

Dość oryginalną koncepcją wyróżnia się algorytm RC (*Relevance in Context*) kontekstowego doboru cech dla rozważanych obiektów, zaproponowany w pracy (Domingos, 1997). Jeśli dana cecha *j* rozważanej próbki *x* jest identyczna (z pewną dokładnością — dobór tej tolerancji jest w pracy uzasadniony) z wartością tej cechy u najbliższego sąsiada *x* i dodatkowo odrzucenie *j*-tej cechy dla próbki *x* nie powoduje zwiększenia błędu szacowanego metodą minus jednego elementu, to *j*-tą cechę dla próbki *x* należy odrzucić. Jak widać, *novum* metody polega na osobnym doborze cech dla poszczególnych obiektów (zauważyć można jednak inspirację omawianymi na początku bieżącego rozdziału kontekstowymi metodami ważenia cech, np. pracą (Aha i Goldstone, 1992)). Na większości testowanych zbiorów UCI, algorytm RC osiągnął wyższą jakość klasyfikacji niż „standardy" FSS i BSS; w porównaniu używano klasyfikatora 1-NN. Można jednak wnioskować, że na zbiorach, dla których optymalny zestaw cech nie zmienia się w zależności od kontekstu, metoda Domingosa nie przyniesie poprawy, a na małych (lub mocno zaszumionych) zbiorach tego typu prawdopodobne jest nawet pogorszenie jakości w stosunku do FSS i BSS, co przyznaje autor metody.

Zastosowanie algorytmów genetycznych do selekcji cech proponowano kilkakrotnie (Brill i in., 1992; Guo i Uhrig, 1992; Vafaie i De Jong, 1992; Cherkauer i Shavlik, 1996; Yang i Honavar, 1997; Kuncheva i Jain, 1999); tylko ostatnia z wymienionych prac dotyczy jednak klasyfikatora 1-NN (w pozostałych schematach klasyfikatorami były drzewa decyzyjne lub sieć neuronowa). Jako ciekawostkę można podać fakt, iż algorytm genetyczny został także z sukcesem wykorzystany do wyboru cech odpowiednich do bezstratnej kompresji chorałów Bacha (Hall, 1995). Za podejściem

genetycznym może przemawiać naturalna reprezentacja cech jako chromosomów (bit 1: dana cecha należy do zestawu, bit 0: cecha nie należy), wewnętrzna równoległość i niestosowanie często wiodącego „na manowce" kryterium monotoniczności oraz elastyczność działania wynikająca z pewnej liczby zadanych z góry współczynników. W pracy Kunchevy i Jaina (1999) połączono selekcję cech z redukcją zbioru odniesienia, osiągając obiecujący kompromis między stopniem redukcji (niskim iloczynem wynikowej liczby cech i liczności zbioru) a jakością. Należy jednak podkreślić, że względna prostota implementacyjna algorytmów genetycznych nie idzie w parze z łatwością zrozumienia ich działania, a zatem wolno chyba stwierdzić, iż eksperymenty w tym kierunku przypominają metodę „prób i błędów".

Bardzo prostą i szybką metodę wyboru zestawu cech o zadanej liczności zaprezentowali Holmes i Navill-Manning (1995). Oceniają oni jakość pojedynczych cech przy użyciu reguł Holte'a (1993), które *de facto* są drzewami decyzyjnymi o jednym poziomie. Użytkownik podaje docelową liczbę cech i otrzymuje na wyjściu odpowiedni zestaw najlepszych (osobno) atrybutów. Niestety, nie ma gwarancji, że cechy najlepsze osobno stanowią najlepszy zestaw łącznie (brak np. zabezpieczenia przed pełną korelacją cech).

Warto jeszcze wspomnieć o równoległym algorytmie konkurencyjnym (ang. *schemata racing*) zaproponowanym w pracy (Maron i Moore, 1997), w którym przyspieszenie procedury otrzymywane jest dzięki odrzucaniu części zbioru odniesienia w trakcie oceny zestawów.

Klasyfikator NNFP

Oryginalna koncepcja została zaprezentowana w pracach (Akkus i Güvenir, 1996) i (Kubat i Chen, 1998). Wszystkie próbki zbioru uczącego są rzutowane na pojedyncze cechy. Klasyfikacja — w schemacie nazwanym regułą najbliższego sąsiada z projekcją cech (*Nearest Neighbor with Feature Projection*, NNFP) — polega na znalezieniu klasy najbliższego sąsiada po każdej poszczególnej cesze, a następnie głosowaniu wyłaniającym zwycięską klasę. W drugiej z wymienionych prac idea została ulepszona. Schemat wNNFP (*Weighted Nearest Neighbor with Feature Projection*) dokonuje heurystycznej redukcji próbek na poszczególnych cechach (bez straty informacji potrzebnej do klasyfikacji), a dodatkowo, co istotniejsze, używa wag dla cech. Cechom, które osobno dość słabo dyskryminują próbki zbioru uczącego, będą przypisane niskie wagi i odwrotnie. Eksperymenty na zbiorach sztucznych i rzeczywistych

potwierdziły to, co można było przypuszczać: *k*-NN daje wyższą jakość w przypadkach małej liczby nieistotnych cech i/lub znaczących korelacji między cechami. Schematy z projekcją cech, a zwłaszcza schemat z wagami, są konkurencyjne w przypadku dużej liczby nieistotnych cech, a także niezależności cech istotnych (warto zauważyć pokrewieństwo klasyfikatora NNFP z naiwnym klasyfikatorem Bayesa, przedstawionym w Rozdz. 1.2.4). Niewątpliwymi zaletami NNFP i wNNFP są szybka klasyfikacja (najbliższy sąsiad dla danej cechy znajdowany jest przy pomocy wyszukiwania binarnego), naturalna obsługa próbek z brakującymi cechami (w takim przypadku dana cecha, innymi słowy: dany klasyfikator składowy, powstrzymuje się od wzięcia udziału w głosowaniu), niewrażliwość na skalowanie cech i pewna odporność na szum (tylko wNNFS).

Poniżej pokazano, jak zastąpić heurystyczną redukcję próbek dla danej cechy redukcją optymalną (idea ta została przeoczona przez Kubata i Chena[5]).

–	+ + +	+ –	– –	–	+	**T_i**
–	+	+ –		–	+	**T_i'**

Rys. 2.2 (przykład z pracy (Kubat i Chen, 1998)). Zbiór T_i' to T_i po redukcji

Heurystyka Kubata i Chena polega na usunięciu wszystkich próbek, dla których najbliżsi sąsiedzi „po obu stronach" są z tej samej klasy, co dana próbka. Ilustruje to Rys. 2.2.

–	+	+ –		–	+	**T_i'**
	o	o		o		**T_i''**

Rys. 2.3. Kółka oznaczają końce następujących po sobie przedziałów,
w których nie zmienia się decyzja o przynależności do klasy

Z kolei **T_i''** na Rys. 2.3 to wynik zastosowania optymalnej redukcji. W procesie klasyfikacji nie szuka się tym razem najbliższego sąsiada dla próbki o wartości danej cechy wynoszącej *x*, natomiast należy znaleźć najbliższy kraniec przedziału (czyli na

[5] M. Kubat, korespondencja prywatna, 2001.

Rys. 2.3 najbliższe kółko) nie mniejszy niż *x*. Następnie odczytuje się z tablicy, z jaką klasą jest ten przedział związany — co kończy klasyfikację dla danej cechy. Warto zauważyć, że w przypadku dwóch klas tablicowanie przedziałów nie jest potrzebne, gdyż wystarczy odczytanie parzystości indeksu danego przedziału (rzecz jasna, aby wyszukiwanie binarne było możliwe, przedziały muszą być przechowywane w posortowanej kolejności).

Rozdział III
Zespoły klasyfikatorów minimalnoodległościowych

W rozdziale przedstawiono koncepcję intensywnie rozwijanych w ostatnich latach zespołów klasyfikatorów, z położeniem szczególnego nacisku na metody dekompozycji oryginalnego zadania dla klasyfikatorów typu *k*-NN i pokrewnych. Opisano opracowaną przez autora rozprawy (Grabowski, 2002a) metodę uczenia zestawów cech dla klasyfikatora „wiele podzbiorów cech" (*Multiple Feature Subsets*, MFS) (Bay, 1998) stosującego głosowanie po różnych podzbiorach cech, wraz z wynikami eksperymentów i ich dyskusją.

3.1. Wprowadzenie

Zasadniczym problemem pojawiającym się praktycznie we wszystkich zadaniach klasyfikacji jest niebezpieczeństwo przeuczenia. Zjawisko to polega na wyborze takiego modelu klasyfikacji, tj. klasyfikatora i jego parametrów, którego bardzo dobra jakość na zbiorze uczącym nie przekłada się na zdolność generalizacji, tj. dobrą predykcję etykiet próbek nieznanych (testowych). Problemu tego nie da się wyeliminować całkowicie, ze względu na skończoną wielkość rzeczywistych zbiorów danych. Duża wariancja błędu w testach wykonywanych na różnych zbiorach uczących i testowych dla danego zadania lub przy kolejnych uruchomieniach algorytmu probabilistycznego sugeruje, iż dany algorytm klasyfikacji jest podatny na przeuczenie.

Jednym z najatrakcyjniejszych podejść mających na celu zwiększenie jakości klasyfikacji i/lub zmniejszenie wariancji błędu jest łączenie klasyfikatorów (ang. *combining classifiers*). Lata 90-te minionego stulecia to okres intensywnych badań w dziedzinie zespołów klasyfikatorów (ang. *ensembles of classifiers*), czego owocem były trzy zasadniczo odmienne podejścia do łączenia klasyfikatorów:

- schematy równoległe z głosowaniem (ang. *classifier fusion*, *mixture of experts*), gdzie poszczególne klasyfikatory składowe dokonują niezależnie predykcji, tj. zwracają etykietę (niekiedy rozmytą) klasyfikowanej próbki, a model wyższego rzędu „scala" ich predykcje (Kittler i in., 1998; Kuncheva i in., 2001) — najpro-

stszą metodą pozyskiwania finalnej decyzji jest głosowanie większościowe (ang. *majority voting*);

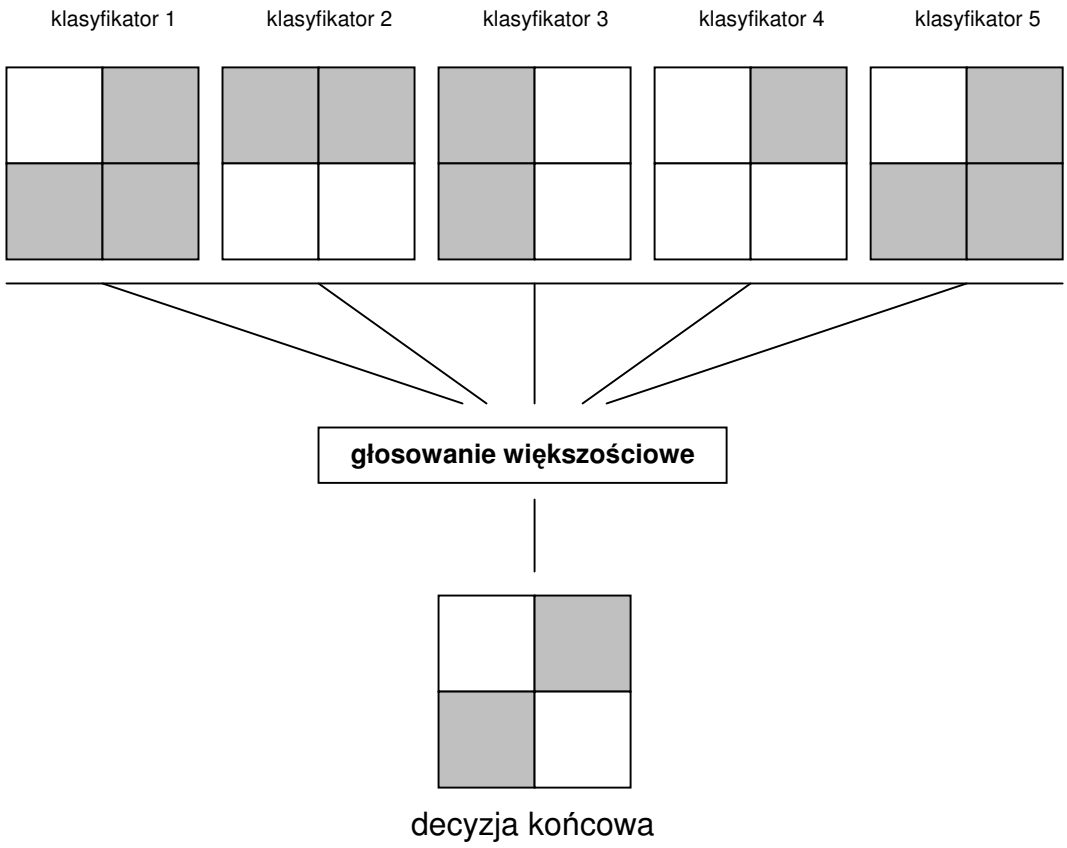
- schematy z lokalnym wyborem klasyfikatora (ang. *classifier selection*), gdzie dla danej próbki wybierany jest odpowiedni (pojedynczy) klasyfikator (Dasarathy i Sheela, 1979);
- schematy kaskadowe (ang. *cascade classifiers*, *multistage classifiers*) — próbka podawana jest na wejście pierwszego, najszybszego, ale i najbardziej „niewyszukanego" klasyfikatora; jeśli uzna on próbkę za zbyt trudną (tj. jego predykcja obciążona byłaby zbyt dużą niepewnością), to odsyła ją do następnego, wolniejszego, ale dokładniejszego klasyfikatora itd. (Baram, 1998; Alpaydin i Kaynak, 1998).

Najwięcej prac dotyczy schematów z głosowaniem. Ideę takiego podejścia można wyjaśnić następująco: jeśli każdy klasyfikator składowy generuje inne przybliżenie prawdziwych granic decyzyjnych między klasami, to można oczekiwać, że w procesie głosowania — czyli uśredniania — szum zostanie (częściowo) usunięty, a granice decyzyjne „wygładzone", zwłaszcza jeśli liczba komponentów jest duża. Istnieje proste twierdzenie mówiące, iż błąd klasyfikacji popełniany w wyniku prostego (większościowego) głosowania zespołu niezależnych klasyfikatorów maleje asymptotycznie do zera, jeśli błędy popełniane przez pojedyncze klasyfikatory są równe i mniejsze od 0,5. Twierdzenie to zostało podane w szeroko cytowanej pracy Hansena i Salamona (1990), warto jednak wiedzieć, iż jest to wynik z XVIII wieku (!), udowodniony przez Markiza Condorceta (Condorcet, 1781). Zdaniem autora niniejszej rozprawy, znaczenie tego twierdzenia jest często przeceniane. Weźmy przykład sztucznego zbioru, dla którego błąd Bayesa można policzyć i niech wynosi on 0,15. Załóżmy teraz, że mamy trzy niezależne klasyfikatory popełniające na tym zbiorze błąd z prawdopodobieństwem 0,2. Proste obliczenia pokazują, że błąd w wyniku głosowania naszego zespołu klasyfikatorów wynosi 0,104 (tj. $0{,}2^3 + 3 \cdot 0{,}2^2 \cdot (1–0{,}2)$). Sprzeczność (tj. „zejście" poniżej błędu Bayesa) pokazuje, iż niemożliwe jest wygenerowanie opisanych trzech niezależnych klasyfikatorów.

Na szczęście, nawet jeśli klasyfikatory nie są niezależne (sytuacja właściwie zawsze pojawiająca się w praktyce), to i tak zwykle można oczekiwać poprawy jakości klasyfikacji w wyniku głosowania. Generalnie, klasyfikatory składowe powinny być możliwie dokładne i możliwie różnorodne (ang. *diverse*), przy czym różnorodność

rozumie się jako możliwie małą korelację błędów poszczególnych klasyfikatorów składowych. Argumentowano również, że negatywna korelacja między klasyfikatorami może być lepsza niż brak korelacji (Kuncheva i in., 2000). Poważną wadą tego podejścia jest jednak wydłużony czas klasyfikacji (w przybliżeniu proporcjonalny do liczby klasyfikatorów składowych i ich średniej szybkości klasyfikacji).

Rys. 3.1 pokazuje, w jaki sposób sieć pięciu prostych klasyfikatorów z głosowaniem większościowym potrafi rozwiązać klasyczny problem „XOR". Warto zwrócić uwagę, iż komponenty sieci nie są niezależne (dla przykładu, klasyfikatory 1 i 5 dokonują zawsze tych samych predykcji).



Rys. 3.1. Realizacja funkcji XOR przy pomocy sieci klasyfikatorów

Charakterystyczną cechą sieci klasyfikatorów z głosowaniem jest zmniejszona zrozumiałość (ang. *comprehensibility*) wygenerowanego modelu w stosunku do klasyfikatora bazowego. W przypadku klasyfikatorów minimalnoodległościowych ma to niewielkie znaczenie, z uwagi na immanentną trudność ludzkiej interpretacji działania nawet pojedynczego klasyfikatora tego typu, jednak w przypadku drzew decyzyjnych — które wszak w najprostszym przypadku dzielą przestrzeń na zestaw

rozłącznych hiperprostopadłościanów płaszczyznami równoległymi do osi układu współrzędnych — może być to znaczną wadą z punktu widzenia końcowego użytkownika takiego systemu.[6]

3.2. Przegląd koncepcji

Poniżej dokonano przeglądu idei tworzenia zespołu klasyfikatorów oraz metod łączenia ich predykcji. W osobnych podrozdziałach omówiono bliżej naszkicowane wcześniej trzy podejścia do konstrukcji klasyfikatorów o strukturze sieciowej, podano metody głosowania klasyfikatorów składowych, a także wspomniano używane miary różnorodności klasyfikatorów w zespole.

3.2.1. Schematy z głosowaniem

Większość schematów wymienionych w tym podrozdziale to klasyfikatory równoległe, tj. z nie komunikującymi się komponentami. Szczególnie w tego typu topologiach sieciowych istotne jest wygenerowanie możliwie różnorodnego zespołu klasyfikatorów składowych.

Bagging (skrót od *bootstrap aggregation*) (Breiman, 1996a) polega na k-krotnym wylosowaniu z powtórzeniami n próbek z n-elementowego zbioru odniesienia, w wyniku czego otrzymywanych jest k zbiorów odniesienia, z których każdy zawiera średnio ok. 63,2% ($=1-1/e$) różnych próbek z oryginalnego zbioru. Każdy z otrzymanych zbiorów służy do wygenerowania osobnego modelu składowego (sieci neuronowej lub drzewa decyzyjnego). W praktyce już 10 składowych drzew decyzyjnych w modelu *bagging* wystarcza do osiągnięcia zadawalającej jakości klasyfikacji (większa liczba komponentów nie daje znaczącej poprawy) (Indurkhya i Weiss, 1998).

Boosting (Schapire, 1990; Freund i Schapire, 1996; Breiman, 1996b) to ogólna metoda przyrostowej konstrukcji zespołu klasyfikatorów (zazwyczaj dla sieci neuronowej), w której każda próbka zbioru uczącego ma swoją wagę, przy czym wagi próbek niepoprawnie klasyfikowanych przez poprzednio zbudowane klasyfikatory są zwiększane, a tym samym nacisk na „nauczenie" się tych właśnie trudnych próbek rośnie. Finalna decyzja powstaje w wyniku ważonego głosowania komponentów. Należy zwrócić uwagę, iż *boosting* nie jest schematem równoległym.

[6] Katastrofa elektrowni w Three-Mile Island — i, być może, w Czernobylu — nie miałaby miejsca, gdyby, mówiąc krótko, ludzie zaufali maszynom (Michie i in., 1994, str. 15).

Breiman (1996b) oraz Schapire i in. (1997) pokazali, iż *bagging* zmniejsza wariancję błędu, czyli poprawia stabilność klasyfikatora. Z drugiej strony *boosting* jest klasyfikatorem mniej przewidywalnym, tj. bardziej wrażliwym na szum (choć średnio osiąga mniejsze błędy niż oryginalny schemat *bagging*). Tłumaczono to zjawisko dwoma przyczynami: nadmiernym być może naciskiem kładzionym na trudne (tj. często zaszumione) próbki (Opitz i Maclin, 1999) oraz ważoną metodą głosowania, która jest bardziej podatna na przeuczenie niż głosowanie większościowe (bez wag) (Sollich i Krogh, 1996).

Zazwyczaj uważa się, że wprowadzenie głosowania z wagami do modelu *bagging* nie przynosi poprawy wyników, z uwagi na zmniejszoną różnorodność komponentów (Indurkhya i Weiss, 1998). Jednak Grossman i Williams (1999) dokonali pomyślnej próby ważenia głosów w tym modelu: zaproponowane wagi uwzględniały jakość predykcji danego komponentu na nie wylosowanych z oryginalnego zbioru próbkach w stosunku do jakości predykcji tych próbek dokonanej przez inne komponenty. Wadą tego podejścia jest większa liczba komponentów potrzebna do stabilizacji wyników (ok. 50), co wydłuża czas uczenia i klasyfikacji oraz zmniejsza zrozumiałość modelu.

Wprowadzanie losowości do modelu jest standardową techniką generacji zespołu sieci neuronowych (Kollen i Pollack, 1991). W najprostszej wersji idea polega na ustawieniu losowych wag startowych dla każdej ze składowych sieci. W literaturze spotyka się sprzeczne stwierdzenia nt. tej metody: wg (Parmanto i in., 1996) efektywność techniki jest stosunkowo niewielka, natomiast w testach na 23 zbiorach UCI oraz z University of Wisconsin wypada ona dobrze, znacząco lepiej niż pojedyncza sieć neuronowa (Opitz i Maclin, 1999).

Eksperymentowano również z dodawaniem losowości do próbek wejściowych. Raviv i Intrator (1996) zakłócali wartości cech szumem gaussowskim, otrzymując tym samym różne zbiory uczące dla zespołu sieci neuronowych. Idea ta, w połączeniu z metodą *bagging*, dała bardzo dobre wyniki na zbiorze sztucznym oraz w zastosowaniu medycznym.

Koncepcję głosowania zespołu klasyfikatorów 1-NN ze zredukowanym zbiorem odniesienia wprowadził Skalak (1997); zostanie ona dokładniej opisana w Rozdz. 6.2. Inną pracę tego typu przedstawił Alpaydin (1997), gdzie zbiorami zredukowanymi były zbiory Harta (p. Rozdz. 6.2), a różnice między nimi wynikały z losowych kolejności próbek zbioru uczącego na wejściu algorytmu redukcji.

Podobnie koncepcje: dekompozycji zadania wielodecyzyjnego na sieć klasyfikatorów binarnych oraz głosowania po różnych zespołach cech zostaną dokładnie omówione w osobnych podrozdziałach (odpowiednio: 3.3 i 3.4–3.5), z uwagi na ich doniosłość dla problematyki klasyfikatorów minimalnoodległościowych.

W Rozdz. 7.4 zaproponowano schemat z głosowaniem wielu klasyfikatorów typu *k*-NN (lub innych klasyfikatorów opartych na *k* sąsiadach), z których każdy znajduje własną wartość *k* na podstawie losowej partycji zbioru uczącego. Eksperymenty pokazują, iż schemat ten osiąga wyższą jakość predykcji niż oryginalna reguła *k*-NN.

3.2.2. Lokalny wybór klasyfikatora

W odróżnieniu od schematów z głosowaniem, schematy z lokalnym wyborem klasyfikatora dla zadanej próbki używają tylko jednego (wybranego) modelu, a zatem są potencjalnie szybsze. Po raz pierwszy koncepcja ta pojawiła się już na przełomie lat 70-tych i 80-tych (Dasarathy i Sheela, 1979; Rastrigin i Erenstein, 1981), jednak nie od razu przebiła się do szerszej świadomości badaczy.[7]

Dasarathy i Sheela (1979) stosują klasyfikator liniowy dla „łatwych" próbek oraz regułę *k*-NN dla próbek „trudnych", zakładając przy tym dwie klasy w zadaniu. Dokładniej, wyznaczana jest para płaszczyzn, z których każda po jednej stronie zawiera próbki tylko z jednej klasy, natomiast pomiędzy płaszczyznami znajduje się tzw. strefa konfliktu (ang. *region of conflict*); próbki testowe, które znajdą się w strefie konfliktu, są klasyfikowane przy użyciu reguły *k*-NN, jednak ze zbiorem odniesienia ograniczonym tylko do próbek uczących z tej strefy.

Rastrigin i Erenstein (1981) używają zestawu klasyfikatorów wraz z *metaklasyfikatorem*, określającym dla danej próbki, w czyj „region kompetencji" ona wpada. Należy zauważyć, iż poprawne przydzielenie modelu dla klasyfikacji danej próbki owocuje klasyfikacją nie gorszą niż oferowana przez dowolny pojedynczy model z danego zespołu użyty globalnie. Rzecz jasna, główna trudność w schematach tego typu polega właśnie na trafnym przydzieleniu modelu.

W pracach (Woods i in., 1997; Giacinto i Roli, 1997) kryterium lokalnego wyboru jest oparte na regule *k* najbliższych sąsiadów z pełnym zbiorem odniesienia, a zatem jest dość wolne (mimo iż użytymi komponentami są perceptrony, a więc klasyfikatory szybsze niż *k*-NN).

[7] W przypadku Rastrigina i Erensteina „zawinił", jak można sądzić, język, w którym artykuł został napisany (rosyjski).

Kuncheva (2000) podzieliła zbiór odniesienia przy pomocy klasteryzacji metodą *k* średnich (MacQueen, 1967), a następnie z każdym klastrem skojarzyła najlepszą dla tego klastra sieć neuronową — jeden z pięciu osobno wyuczonych perceptronów wielowarstwowych. Głównym wnioskiem, jaki można wyciągnąć z wyników autorki, jest możliwość uzyskania mniejszych błędów przy lokalnym wyborze zbioru odniesienia niż przy głosowaniu większościowym zespołu, gdy składowe sieci neuronowe są „niedouczone" (mała liczba epok w sesjach treningowych). Oznacza to, że zaletą schematu z lokalnym wyborem zbioru odniesienia może być nie tylko szybka klasyfikacja, ale także stosunkowo krótki czas uczenia.

Warto dodać, że istnieją algorytmy pośrednie pomiędzy podejściem pierwszym (głosowanie) a drugim (lokalny wybór jednego modelu). Mianowicie w pracach (Jacobs i in., 1991) oraz (Alpaydin i Jordan, 1996) autorzy nominują lokalnie do podjęcia decyzji pewną grupę klasyfikatorów, które następnie uczestniczą w głosowaniu.

W Rozdz. 6.4 zaproponowano schemat z lokalnym wyborem zbioru zredukowanego, gdzie kryterium wyboru może być oparte na klasteryzacji lub podziale hiperpłaszczyznami zbioru uczącego.

3.2.3. Schematy kaskadowe

Komponenty w zespole nie muszą być klasyfikatorami tego samego typu ani podobnej złożoności. W praktyce, rozpiętość kosztu klasyfikacji pomiędzy najprostszymi i najszybszymi a najwolniejszymi używanymi klasyfikatorami, np. między klasyfikatorem liniowym a *k*-NCN (p. Rozdz. 7), wynosi nawet 4–5 rzędów wielkości. Rzecz jasna, klasyfikatory prostsze zwykle oferują gorszą predykcję, gdyż nie uwzględniają pełnej złożoności problemu.

Koncepcja klasyfikatora kaskadowego bazuje na powyższej obserwacji. Uczenie polega na wygenerowaniu kilku (w praktyce zwykle zaledwie 2–3) klasyfikatorów uszeregowanych od najprostszego do najbardziej złożonego. Oprócz samych klasyfikatorów potrzebne są też kryteria ufności, tj. oceniające, czy dany etap klasyfikacji jest wystarczający do dostatecznie pewnej predykcji zadanej próbki. Jeśli pierwszy klasyfikator w zespole nie potrafi nadać etykiety testowej próbce z dostateczną pewnością, to próbka zostaje przekazana na wejście drugiego klasyfikatora. Analogiczna reguła rządzi również na następnych etapach klasyfikacji. W przypadku najprostszym, tj. gdy klasyfikator kaskadowy złożony jest tylko z dwóch etapów, etap pierwszy może być postrzegany jako ogólna reguła, natomiast etap drugi jest obsługą wyjątków od tej

reguły („trudnych" próbek). Zakładając, że klasyfikacja większości próbek zostaje zakończona na pierwszym, szybkim, etapie, klasyfikator kaskadowy może być atrakcyjnym rozwiązaniem zapewniającym zarówno wysoką jakość, jak i szybkość predykcji. W praktyce, oczywiście, nie jest łatwo zarówno dobrać zespół klasyfikatorów, jak i kryteria ufności.

Kaynak i Alpaydin (2000) przedstawili klasyfikator dwu- lub trójstopniowy używający jednego lub dwóch perceptronów wielowarstwowych (lub perceptronu jednowarstwowego w pierwszej fazie) oraz reguły *k*-NN w ostatniej fazie klasyfikacji, osiągając na kilku zbiorach UCI jakość porównywalną z jakością zwykłej reguły *k*-NN przy kilkakrotnie wyższej szybkości klasyfikacji.

Jóźwik i in. (1996) zaproponowali klasyfikator kaskadowy używający reguł 1-NN i *k*-NN. Jeśli próbka testowa jest położona dostatecznie blisko obiektów jednej tylko klasy, to zostaje ona przypisana do tej klasy, a zatem zgodnie z regułą 1-NN. W sytuacji, gdy próbka testowa znajduje się w strefie nakładania się klas (co w najgorszym przypadku można stwierdzić po policzeniu odległości do wszystkich próbek zbioru uczącego, a więc po wykonaniu pierwszej fazy klasyfikacji), o jej etykiecie decyduje reguła *k*-NN. Wreszcie w przypadku, gdy dana próbka położona jest poza strefami wszystkich klas, klasyfikator wstrzymuje się od podjęcia decyzji. Taka konstrukcja klasyfikatora może być przydatna w aplikacjach z mocno niesymetryczną funkcją straty, np. w medycynie, gdzie sklasyfikowanie pacjenta chorego jako zdrowego może być znacznie groźniejsze niż pomyłka odwrotna.

W Rozdz. 7.5 zaproponowano rodzinę klasyfikatorów kaskadowych, wykorzystującą koncepcję symetrycznego sąsiedztwa oraz podejmującego decyzję w pierwszym etapie, jeśli istnieje konsensus wśród klasyfikatorów równoległych tego etapu.

3.2.4. Metody łączenia głosów klasyfikatorów równoległych

Większość równoległych schematów klasyfikacji może być postrzegana jako metaklasyfikatory: pierwszy (najniższy) poziom to osobne jednostki działające na oryginalnych danych, poziom wyższy zaś to klasyfikator, którego danymi są wyjścia klasyfikatorów pierwszego poziomu (*stacked generalization*, Wolpert, 1992). W najprostszej wersji klasyfikator drugiego poziomu jest trywialny: dokonuje on tylko głosowania większościowego, tj. przypisuje próbkę do tej klasy, na którą oddało głos najwięcej klasyfikatorów składowych pierwszego poziomu. W przypadku, gdy klasyfikatory składowe zwracają rozmyte etykiety klas, możliwe są także na ich wyjściach operacje:

„maksimum", „minimum", uśrednienie oraz iloczyn (wszystkie wspomniane metody opisano np. w pracy (Kittler i in., 1998)).

Metody powyższe nie wymagają uczenia. Istnieją jednak schematy łączenia głosów składowych adaptujące się do zbioru uczącego, które w pełni zasługują na określenie ich mianem klasyfikatora. Przykładowo, metoda *behavior-knowledge space* (BKS) (Huang i Suen, 1995) polega na znalezieniu dla każdej możliwej kombinacji wyjść zespołu L klasyfikatorów składowych najczęstszej etykiety klasy wśród tych elementów zbioru uczącego, dla których zespół klasyfikatorów zwraca daną L-elementową kombinację odpowiedzi. Próbka x zbioru uczącego jest przypisywana do klasy skojarzonej z zespołem odpowiedzi klasyfikatorów składowych dla x (obsługa przypadków kombinacji odpowiedzi, które nie wystąpiły dla żadnego elementu zbioru uczącego, jest osobnym zagadnieniem).

Innymi metodami głosowania wymagającymi uczenia są m. in. *kombinacja Bayesa* (ang. *Bayes combination*) (Xu i in., 1992) i użycie wzorców decyzyjnych (ang. *decision templates*) (Kuncheva i in., 2001).

3.2.5. Miary korelacji klasyfikatorów składowych

Innym zagadnieniem jest pomiar korelacji — lub „różnorodności" (ang. *diversity*) — zespołu klasyfikatorów składowych. Ponieważ prawdziwa niezależność komponentów jest w praktyce niemożliwa, pożądaną rzeczą byłaby choć znajomość korelacji popełnianych przez nie błędów, celem np. odrzucenia komponentów zbyt „podobnych" do innych, które pogarszają predykcję i/lub szybkość działania całego układu. Estymacja korelacji nie jest łatwa, z uwagi na niewielkie zazwyczaj liczności zbiorów uczących.

Należy podkreślić, że miary korelacji komponentów zespołu klasyfikatorów niekoniecznie są bardziej wiarygodne od bezpośredniej estymacji błędu zespołu na zbiorze uczącym. Ich wprowadzanie argumentuje się raczej możliwością głębszego zrozumienia, jak działa klasyfikator równoległy, co — można oczekiwać — pomoże w znalezieniu lepszych heurystyk lub nawet matematycznej teorii projektowania efektywnych zespołów.[8]

Ogólnie miary korelacji można podzielić na dwie kategorie: estymujące ją na bazie wszystkich par klasyfikatorów składowych i uśredniające wyniki (ang. *pairwise*

[8] L. Kuncheva, korespondencja prywatna, 2002.

diversity measures) oraz w bezpośredni sposób estymujące korelację dla całego zespołu (ang. *non-pairwise diversity measures*).

Do miar pierwszej grupy, z których wiele znanych jest w statystyce od dziesięcioleci (Sneath i Sokal, 1973), można zaliczyć m. in. współczynnik Yule'a Q przyjmujący wartości od –1 (maksymalna negatywna korelacja) przez 0 (niezależność) do 1 (pełna pozytywna korelacja, tj. oba klasyfikatory popełniają błędy na dokładnie tych samych próbkach), miarę niezgodności D, tj. frakcję próbek ze zbioru uczącego, dla których jeden z pary klasyfikatorów dawał poprawną odpowiedź, a drugi błędną, oraz miarę „podwójny błąd" (ang. *double-fault measure*), czyli frakcję próbek, dla których oba klasyfikatory podały błędną odpowiedź. Wszystkie wymienione miary opisane są w cytowanej wyżej monografii Sneatha i Sokala.

Do miar w bezpośredni sposób estymujących korelację całego zespołu należą m. in. miara trudności θ (ang. *measure of difficulty*) (Hansen i Salamon, 1990), będąca wariancją frakcji poprawnych odpowiedzi komponentów dla poszczególnych próbek zbioru uczącego oraz wariancja Kohaviego–Wolperta (1996), oparta na podobnych przesłankach, lecz niejako faworyzująca (w sensie niższej estymowanej korelacji) komponenty o predykcji bliskiej 50% (są to często tzw. słabe (ang. *weak*) klasyfikatory, czyli klasyfikatory o zdolności predykcji niewiele wyższej od losowego odgadywania).

Wszystkie wymienione metody łączenia głosów oraz miary korelacji komponentów zostały eksperymentalnie zbadane przez Kunchevą i in. (Shipp i Kuncheva, 2002; Kuncheva i Whitaker, 2003); na podstawie tych prac oraz pracy (Kuncheva, 2002), do najlepszych metod łączenia głosów klasyfikatorów składowych należy zaliczyć wzorce decyzyjne, kombinację Bayesa i prawdopodobnie głosowanie większościowe, natomiast miarami korelacji najbardziej wrażliwymi na wybór metody głosowania, czyli mogącymi znacząco pomóc na etapie projektowania klasyfikatora, są miara trudności θ oraz nieco gorsza, lecz szybsza obliczeniowo, miara „podwójny błąd".

3.3. Dekompozycja zadania wielodecyzyjnego na sieć podzadań binarnych

W przypadku złożonych zadań klasyfikacji można spodziewać się sukcesu przy właściwym zastosowaniu algorytmów działających w oparciu o zasadę „dziel i rządź" (ang. *divide-and-conquer*). Niejednokrotnie w praktyce liczba rozróżnianych klas

w zadaniu jest większa od dwóch. Można zatem oczekiwać, że podejście polegające na rozłożeniu zadania wielodecyzyjnego na równoległą sieć klasyfikatorów binarnych (dychotomizerów), a następnie złożeniu ich wyników w celu podjęcia finalnej decyzji, umożliwi dokładniejsze skupienie się na podzadaniach i w konsekwencji doprowadzi do poprawy jakości klasyfikacji.

Schemat dekompozycyjny zadania wielodecyzyjnego ma dwa etapy. W pierwszym dana próbka jest podawana (równolegle) na wejście wielu osobnych (tj. nie komunikujących się) klasyfikatorów dwudecyzyjnych. W drugim etapie nadawana jest próbce etykieta klasy na podstawie wyjść komponentów sieci.

Wady i zalety takiego podejścia są w dużej mierze analogiczne do wad i zalet innych schematów równoległych. Zasadniczą wadą jest arbitralność wyboru komponentów i metody głosowania (scalania głosów składowych). Komponenty dwudecyzyjne na ogół działają z niekompletną informacją względem pełnego (oryginalnego) zbioru danych. Utrata informacji polega zazwyczaj na odrzuceniu części próbek (np. ograniczeniu się tylko do dwóch danych klas) lub nadania nowych etykiet elementom zbioru uczącego tak, aby zredukować liczbę klas do dwóch (można w takim przypadku użyć terminu: „superklasy"). Dodatkową wadą podejścia dekompozycyjnego jest zazwyczaj wolniejsza, w stosunku do modelu bazowego, klasyfikacja i uczenie.

Zaletą podejścia jest natomiast osobne uczenie stosowane do klasyfikatorów składowych, np. osobna selekcja cech i osobne szukanie optymalnej wartości k dla k-NN. Co więcej, niektóre typy klasyfikatorów mogą z natury być przeznaczone do zadań binarnych (np. szukające liniowej rozdzielności klas czy ich podzbiorów), a ich uogólnienie na wiele klas nie jest proste, a zatem schemat dekompozycyjny cechuje się dużą elastycznością. Dodatkową zaletą podejścia są własności samokorygujące (ang. *self-correcting*) schematu równoległego. Oznaczają one, że pojedynczy klasyfikator może się mylić, ale prawdopodobieństwo, że błędną decyzję podejmie zespół (np. w wyniku prostego głosowania) powinno być mniejsze, gdyż mylne głosy często zostaną przegłosowane.

Schematy dekompozycyjne można podzielić na dwie grupy: te, w których dekompozycja dokonywana jest *a priori*, tj. bez uwzględniania informacji płynących ze zbioru uczącego, oraz schematy *a posteriori*, w których topologia sieci powstaje po zapoznaniu się ze zbiorem uczącym. Nietrudno odgadnąć, że schematy z dekompozycją *a posteriori* są bardziej obiecujące.

3.3.1. Algorytm „jedna klasa przeciwko pozostałym”

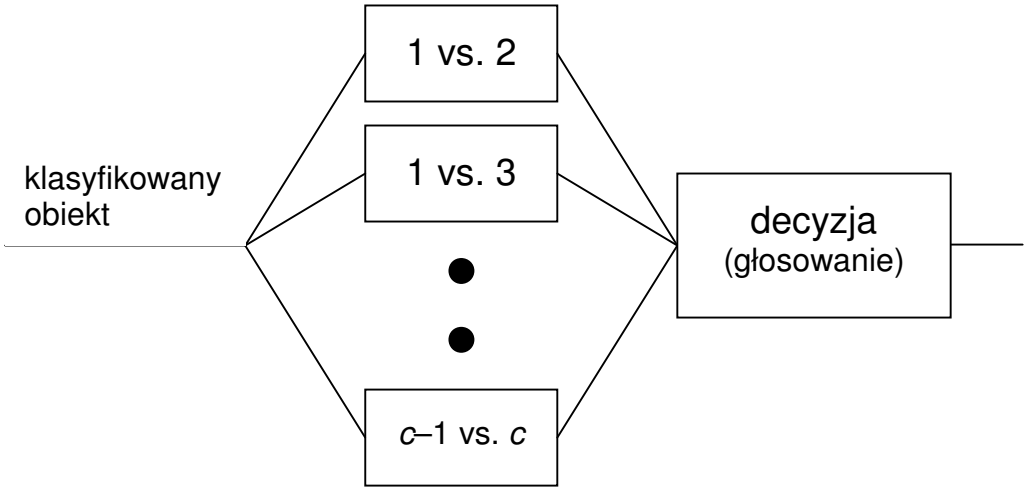
Zapewne najprostszym schematem dekompozycji zadania wielodecyzyjnego jest algorytm „jedna klasa przeciwko pozostałym” (ang. *One-Per-Class*, OPC) (Nilsson, 1965). W oryginalnej propozycji każda z c działających równolegle sieci neuronowych uczyła się indywidualnej funkcji binarnej f_i, $i = 1..c$, skojarzonej z i-tą klasą. Klasyfikowana próbka p jest przypisywana do tej klasy j, dla której funkcja f_j zwraca największą wartość. W przypadku klasyfikatorów minimalnoodległościowych komponentami OPC są klasyfikatory k-NN wybierające pomiędzy pojedynczą klasą (inną dla każdego z c komponentów) a superklasą powstałą ze scalenia $c-1$ pozostałych klas. W wyniku finalnej decyzji nieznany obiekt przypisany zostaje do tej z oryginalnych klas, która uczestniczyła w największej liczbie zwycięstw. Łatwo widać, że OPC jest schematem *a priori*.

W pracy (Moreira i Mayoraz, 1998) porównano kilka schematów dekompozycyjnych, gdzie klasyfikatorami składowymi były drzewa decyzyjne C4.5. Schemat OPC wypadł najsłabiej pod względem jakości, nieraz gorzej od oryginalnego klasyfikatora wielodecyzyjnego. Jest to zrozumiałe, gdy zauważy się, że zdolności samokorygujące tego schematu są zerowe. Pomyłka tylko jednego komponentu już może prowadzić do błędnej finalnej decyzji.

3.3.2. Algorytm Jóźwika–Vernazzy

Lepszą koncepcją jest dekompozycja zadania na sieć dychotomizerów, z których każdy rozróżnia tylko pomiędzy parą klas (ang. *Pairwise Coupling*, PWC). Ponieważ par klas jest $c \cdot (c-1)/2$, więc tyleż jest komponentów sieci. Decyzja wynikowa powstaje w procesie prostego głosowania. Schemat ten (Rys. 3.2), używany od dawna dla perceptronów wielowarstwowych (Duda i Hart, 1973), w kontekście klasyfikatorów minimalnoodległościowych został zastosowany po raz pierwszy w pracy (Jóźwik i Vernazza, 1988).

Schemat PWC został sprawdzony w szeregu prac, m. in. (Moreira i Mayoraz, 1998; Jóźwik, 2001), które dowiodły jego efektywności. W pracy (Grabowski, 2001b) użyto tego schematu dla klasyfikatora k-NCN, również otrzymując poprawę jakości w stosunku do oryginalnej reguły k-NCN. Wyniki eksperymentów autora rozprawy z dekompozycją PWC są zamieszczone w Rozdz. 7.6.



Rys. 3.2. Ogólny schemat dekompozycji PWC zadania *c*-decyzyjnego

3.3.3. Algorytm Moreiry–Mayoraza

Wadą schematu PWC jest przekazywanie do drugiego etapu (tj. do głosowania) nonsensownych odpowiedzi klasyfikatorów binarnych rozróżniających między pewnymi klasami *i* oraz *j* w przypadku, gdy wejściowa próbka nie należy ani do klasy *i*, ani do klasy *j*. Co prawda można oczekiwać, że „odpowiednie" klasyfikatory składowe zazwyczaj „przegłosują" takie przypadkowe głosy, niemniej mamy tu do czynienia z szumem, który w jakimś stopniu negatywnie wpływa na jakość klasyfikacji.

Dość udaną próbą naprawienia tego defektu był algorytm „skorygowany zestaw par klas" (ang. *Pairwise Coupling — Correcting Classifiers*, PWC-CC) zaproponowany w pracy (Moreira i Mayoraz, 1998). Dla danej próbki *p* i danego klasyfikatora składowego rozróżniającego między klasami *i* oraz *j* przeprowadzana jest także pomocniczo klasyfikacja *p* klasyfikatorem binarnym rozróżniającym między superklasą złożoną z sumy mnogościowej klas *i* oraz *j*, a superklasą złożoną ze wszystkich pozostałych klas. Wynikiem opisanej pomocniczej klasyfikacji jest liczba rzeczywista z przedziału [0, 1], która będzie stanowiła wagę głosu „oryginalnego" klasyfikatora składowego rozróżniającego między parą klas (*i*, *j*). Jeśli klasyfikator pomocniczy ma zwracać decyzję ostrą (0 lub 1), to *novum* schematu PWC-CC można streścić następująco: jeśli dana próbka wydaje się należeć do klasy *i* lub *j*, to wtedy i tylko wtedy głos klasyfikatora „*i* kontra *j*" jest uwzględniany.

Jak widać, PWC-CC jest, w przeciwieństwie do dwóch poprzednich algorytmów, schematem *a posteriori*. W cytowanej pracy przewaga jakości predykcji schematu PWC-CC nad PWC była dość znaczna na kilku zbiorach UCI, przy zastosowaniu drzew

decyzyjnych C4.5 jako dychotomizerów. Również praca (Masulli i Valentini, 2000) potwierdza efektywność schematu, tym razem z perceptronami MLP jako składowymi sieci. W Rozdz. 7.6 zamieszczono wyniki autora rozprawy dla schematu PWC-CC wykorzystanego w klasyfikatorach minimalnoodległościowych.

3.3.4 Wyjściowe kody samokorygujące (ECOC)

Inną koncepcją schematu dekompozycji było rozważenie „sztucznego" zestawu klasyfikatorów dwudecyzyjnych, mając na uwadze trzy kryteria: łańcuchy binarne („słowa kodowe", ang. *code words*) dla różnych klas muszą być maksymalnie odległe w sensie metryki Hamminga; odległości między parami słów kodowych mają być możliwie równe i wreszcie poszczególne klasyfikatory składowe muszą się możliwie różnić przy klasyfikacji różnych klas, gdyż tylko taka różnorodność zapewnia małą korelację między nimi.

Konkretną propozycją utworzenia takiego zestawu słów kodowych był schemat z „wyjściowymi kodami samokorygującymi" (*Error-Correcting Output Codes*, ECOC) zaproponowany przez Diettericha i Bakiriego (1991). Jako słowa kodowe przyjęto kody samokorygujące BCH (Bose i Ray-Chauduri, 1960), co zagwarantowało utrzymanie minimalnej odległości Hamminga d między słowami (a zatem schemat ECOC potrafi skorygować maksymalnie $\left\lfloor \frac{d}{2} \right\rfloor$, d – nieparzyste, błędów klasyfikatorów składowych). W kolejnej pracy tych autorów (Dietterich i Bakiri, 1995), słów kodowych szukano przy pomocy stochastycznej wersji procedury „wspinania się po wzgórzu" (ang. *hill climbing*). W obu pracach stwierdzono, iż ECOC zwiększa jakość klasyfikacji przy klasyfikatorach działających globalnie (drzewa decyzyjne, np. ID3, C4.5, sieci neuronowe z propagacją wsteczną), natomiast uważano, że schemat ten nie sprawdza się przy klasyfikatorach lokalnych (np. k-NN), z uwagi na dużą korelację między składowymi dychotomizerami. Ricci i Aha (1998) pokazali jednak, że osobna selekcja cech dla poszczególnych klasyfikatorów składowych umożliwia osiągnięcie sukcesu w tym podejściu także przy klasyfikatorach typu 1-NN, o ile dla różnych komponentów zostaną wybrane różne zestawy cech. Co więcej, w cytowanej pracy potwierdzono eksperymentalnie intuicyjne przypuszczenie, iż schemat ECOC przynosi korzyść w tym większym stopniu, im większa jest różnorodność pomiędzy wybranymi zestawami cech. W pracy stwierdzono także, że poprawa jakości w schemacie NN-ECOC okupiona jest nieznacznym wzrostem wariancji wyników.

W pracy (Windeatt i Ghaderi, 2000) pokazano, iż z założenia optymalności (w sensie Bayesa) klasyfikatorów składowych (dychotomizerów działających na superklasach) i jednakowej odległości Hamminga między wszystkimi parami słów kodowych wynika, iż najlepszą możliwą strategią klasyfikacji — równoważną regule Bayesa — jest wybór klasy najbliższej w sensie odległości Hamminga między odpowiednimi słowami kodowymi.

3.4. Równoległe podejście do problemu selekcji cech

Klasyfikatory minimalnoodległościowe są stabilne, tzn. dodanie lub usunięcie niewielkiej liczby próbek do/ze zbioru uczącego w niewielkim stopniu wpływa na predykcję. Cecha ta odróżnia metodę *k*-NN (oraz 1-NN z pełnym zbiorem odniesienia) od np. drzew decyzyjnych.

Stabilność klasyfikatora sprawia, iż niełatwo jest wygenerować zespół istotnie różnych (ang. *diverse*) klasyfikatorów tego typu. Jeśli w zespole nie ma dostatecznej różnorodności, wówczas trudno oczekiwać poprawy klasyfikacji przy kolektywnej predykcji. Argumentowano, iż technika *bagging*, opisana w Rozdz. 3.2, „nie działa" w połączeniu z regułą *k*-NN.

Sytuacja nie jest jednak beznadziejna. Jedną z kilku możliwości wygenerowania różnorodnego zespołu klasyfikatorów minimalnoodległościowych jest użycie różnych zestawów cech dla komponentów zespołu. Idea znana jest od kilku lat w kontekście sieci neuronowych (Tumer i Ghosh, 1996; Cherkauer, 1996), natomiast pierwszymi doniesieniami o użyciu różnych zestawów cech w zespołach klasyfikatorów minimalnoodległościowych były powstałe niezależnie prace (Bay, 1998) i (Ho, 1998b). W tym samym okresie Ho opublikowała jeszcze jeden artykuł zawierający opis analogicznej techniki zastosowanej do drzew decyzyjnych (Ho, 1998a). Co ciekawe, w pracach Bay'a i Ho zestawy użytych cech są losowe.

Bay określił swój algorytm etykietą: „wiele podzbiorów cech" (*Multiple Feature Subsets*, MFS). Ten klasyfikator równoległy z innego punktu widzenia jest *de facto* niekonwencjonalnym podejściem do problemu selekcji cech. Zamiast wyboru pojedynczego zestawu „dobrych" cech, klasyfikator MFS wybiera wiele (w eksperymentach Bay'a aż 100) czysto losowych zestawów cech; liczba cech w każdym zestawie jest jednakowa i zostaje ustalona dla zbioru uczącego przy pomocy walidacji skrośnej lub metody minus jednego elementu zastosowanej z użyciem całego zespołu jako

estymatora błędu klasyfikacji. Klasyfikacja nieznanych próbek polega na głosowaniu większościowym klasyfikatorów składowych typu 1-NN, z których każdy uwzględnia jedynie własny zestaw cech. Bay testował dwie wersje algorytmu: z losowaniem cech dla zestawu z powtórzeniami i bez powtórzeń; ponieważ jednak wyniki obu wersji były bardzo zbliżone, autor niniejszej rozprawy postanowił w testach ograniczyć się tylko do bardziej konwencjonalnej wersji bez powtórzeń. Stwierdzono, że klasyfikator jest dość silnie wrażliwy na liczbę cech w zestawie; jeżeli jednak liczba ta jest wybrana przy pomocy zbioru uczącego (a nie zadana *ad hoc*), wówczas w średnim przypadku można oczekiwać, iż MFS osiągnie wyższą jakość niż takie klasyczne metody selekcji cech jak FSS i BSS, a co więcej, będzie także bardziej odporny na szum.

3.5. Uczenie zestawów cech dla klasyfikatora MFS

W pracy (Grabowski, 2002a) autor rozprawy dokonał modyfikacji wylosowanych zbiorów cech dla klasyfikatora równoległego MFS. Stochastyczny charakter procedury selekcji podzbiorów cech, którą opisano poniżej, sugeruje iż otrzymane klasyfikatory składowe nadal będą „do pewnego stopnia" niezależne, podczas gdy ich indywidualne zdolności predykcji zostaną znacząco polepszone w stosunku do zestawów losowych.

Procedura selekcji zestawów cech, którą stosowano, została zainspirowana metodą generacji zbioru zredukowanego Skalaka (Skalak, 1994). Jest to prosta technika probabilistyczna, która w niniejszej wersji wygląda następująco. Na wstępie losowanych jest r cech z pełnego zbioru (r – zadane z góry). Następnie w pętli próbuje się dokonywać tzw. mutacji, polegających na odrzuceniu jednej losowej cechy z aktualnego zestawu wybranych cech i jednoczesnym dodaniu jednej losowej cechy ze zbioru wszystkich cech aktualnie nie wybranych. Jeśli wynikiem takiej zamiany jest poprawa jakości klasyfikacji na zbiorze uczącym, estymowana metodą minus jednego elementu, wtedy i tylko wtedy zamianę (=mutację) należy zaakceptować. Iteracja taka powtarzana jest m razy, gdzie m jest kolejnym wolnym parametrem.

Opisana procedura wykonywana jest L-krotnie i w wyniku otrzymuje się L klasyfikatorów, które będą uczestniczyły w głosowaniu.

Przeprowadzono eksperymenty na parach klas zbioru Ferrites (30 cech). Użyto 10 niezrównoważonych partycji całego zbioru i dla każdej partycji (tj. zbioru uczącego i testowego) w danym teście używano tylko jednej wybranej pary klas. Ponieważ 6 z 8 klas w 1400-elementowych oryginalnych zbiorach uczących liczyło po 200 obiektów,

a dwie klasy po 100 obiektów, zatem zbiory uczące w rozważanych zadaniach binarnych liczyły od 200 do 400 obiektów. Tab. 3.1 prezentuje uśrednione po 10 uruchomieniach wyniki (błędy predykcji) dla każdej pary klas oraz czterech algorytmów.

Tabela 3.1: Głosowanie zespołów cech
kontra klasyczne strategie selekcji cech (BSS i FSS)

para klas	błąd [%]			
	BSS	FSS	MFS, zbiory losowe (5 • 5)	zbiory otrzymane dzięki RMHC (5 • 5)
1/2	0,81 (5,0)	0,74 (2,4)	1,81	**0,64**
1/3	0,33 (1,6)	**0,10** (1,0)	0,45	0,21
1/4	0,29 (3,1)	0,50 (1,8)	2,53	**0,27**
1/5	0,16 (1,9)	0,05 (1,0)	0,10	**0,03**
1/6	8,06 (12,0)	**6,03** (16,3)	10,82	6,73
1/7	0,50 (1,3)	**0,07** (1,1)	0,42	0,19
1/8	9,42 (10,4)	9,05 (13,8)	8,94	**8,54**
2/3	3,05 (8,5)	2,90 (6,4)	3,40	**2,16**
2/4	3,47 (9,5)	3,02 (6,7)	5,85	**2,97**
2/5	0,64 (3,1)	**0,32** (2,7)	0,75	0,51
2/6	0,44 (5,0)	0,59 (5,3)	1,63	**0,38**
2/7	4,10 (15,5)	2,44 (10,3)	4,24	**2,05**
2/8	1,18 (6,7)	0,96 (3,7)	1,79	**0,61**
3/4	1,73 (6,4)	1,04 (3,1)	2,47	**0,78**
3/5	1,80 (5,3)	1,76 (3,6)	2,21	**1,59**
3/6	0,19 (1,0)	0,32 (1,0)	0,52	**0,07**
3/7	9,03 (12,3)	8,70 (16,0)	11,28	**8,14**
3/8	0,26 (1,2)	0,23 (1,3)	0,76	**0,18**
4/5	3,69 (9,4)	3,79 (12,1)	3,47	**3,46**
4/6	**0,14** (2,1)	0,15 (1,2)	1,87	0,15
4/7	1,02 (5,0)	0,87 (3,1)	1,12	**0,54**
4/8	0,12 (3,8)	0,16 (1,5)	2,09	**0,09**
5/6	0,36 (2,3)	**0,05** (1,0)	0,33	0,06
5/7	2,31 (5,6)	**2,25** (7,9)	2,28	2,30
5/8	0,35 (1,6)	**0,07** (1,0)	0,27	0,19
6/7	0,52 (2,7)	0,35 (1,3)	1,12	**0,21**
6/8	23,68 (17,2)	19,93 (12,6)	19,99	**16,86**
7/8	1,02 (4,8)	0,59 (1,6)	2,42	**0,54**

Przetestowano:

- strategię usuwania cech (BSS);
- strategię dodawania cech (FSS);
- MFS z $L = 5$ zestawami losowych cech (bez powtórzeń), każdy liczący $r = 5$ cech;
- opisaną propozycję z $L = 5$ znalezionymi zestawami cech i $m = 200$; i tu każdy zestaw liczył $r = 5$ cech.

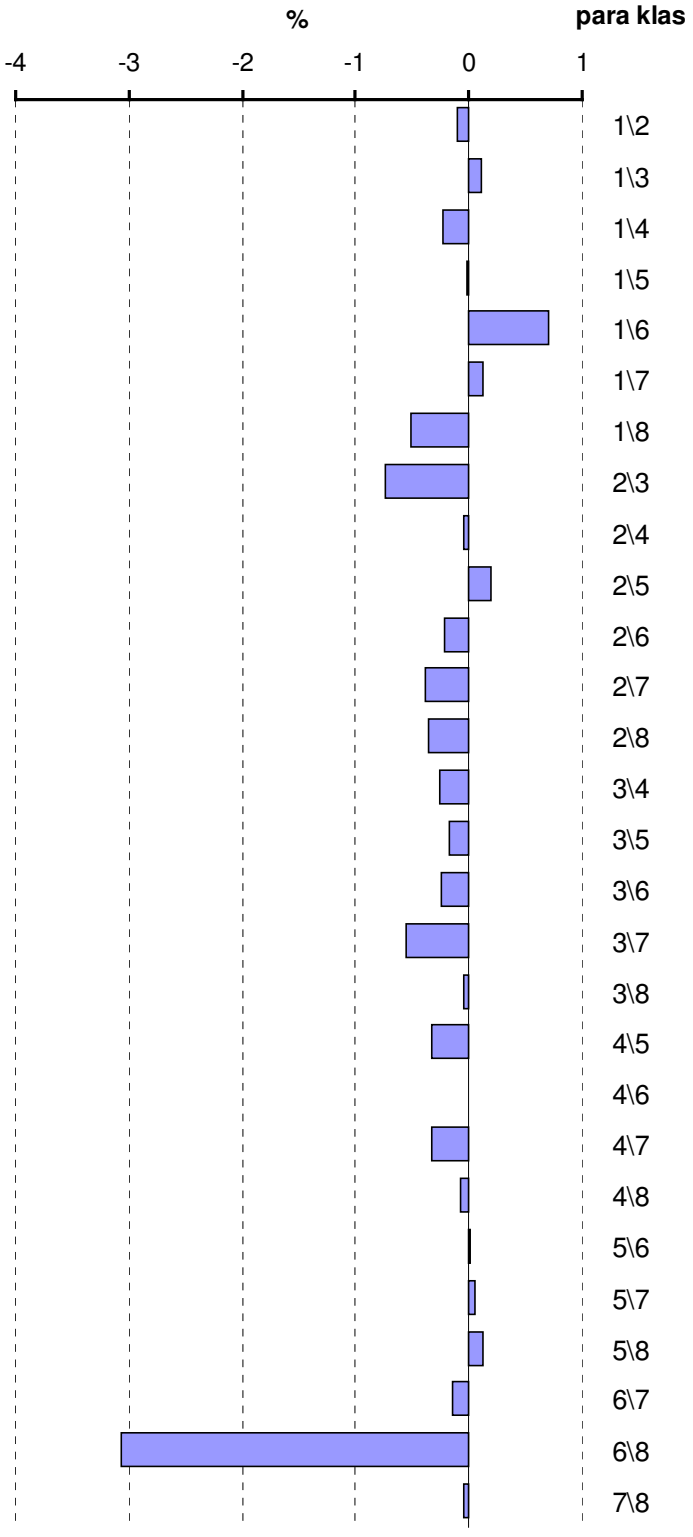
Wartości *L*, *r* i *m* zostały zadane *ad hoc*. Liczby w nawiasach w kolumnach FSS i BSS to średnie liczności wyselekcjonowanych zbiorów cech.

Jak wynika z Tab. 3.1 (wynik najlepszej metody dla każdej pary klas został pogrubiony), w 20 z 28 przypadków prezentowana metoda (ostatnia kolumna; skrót RMHC oznacza „random mutation hill climbing") pokonała algorytmy konkurencyjne. FSS dominowała siedmiokrotnie, zaś BSS tylko raz. Zwykły klasyfikator MFS w większości przypadków oferował zdecydowanie niższą jakość predykcji od pozostałych metod. Rzecz jasna, brak uczenia (tj. wyboru liczności podzbiorów) i zbyt mała liczba komponentów w oryginalnym MFS były główną przyczyną tej klęski, jednak czynniki te obniżają efektywność również wprowadzonego schematu. Mimo to, proponowany algorytm typu RMHC osiągnął w większości przypadków wyższą jakość klasyfikacji niż standardowe techniki FSS i BSS.

Na Rys. 3.3 przedstawiono w postaci graficznej różnice błędów klasyfikacji między zaproponowaną metodą a strategią FSS dla poszczególnych par klas zbioru Ferrites (wartości ujemne oznaczają mniejszy błąd proponowanego algorytmu).

Interesującym jest pytanie, dlaczego strategia BSS wypadła tak słabo, w szczególności: przeważnie gorzej niż FSS. Poniżej podjęto próbę wytłumaczenia tego zjawiska.

Jeśli oryginalna liczba cech jest duża („duża" zawsze oznacza: „duża w stosunku do liczności zbioru"), to można przypuszczać, że optymalna selekcja cech powinna wybrać niewielką część oryginalnego zbioru, bo większy zestaw prowadzi zazwyczaj do przeuczenia. Jeśli zatem, mówiąc mało precyzyjnie, optymalna liczba cech jest mniejsza od ok. połowy oryginalnej liczby cech, to FSS (przypomnijmy: zaczynająca działanie od pustego zestawu) ma do przebycia (tj. do znalezienia optimum) krótszą drogę niż BSS. Na dłuższej drodze zaś łatwiej wybrać niewłaściwą ścieżkę.



Rys. 3.3. Różnice bezwzględnych błędów klasyfikacji (w %) między zaproponowaną metodą selekcji cech a strategią FSS dla poszczególnych par klas zbioru Ferrites.
Wartości ujemne oznaczają poprawę jakości klasyfikacji.

W wielu eksperymentach obserwuje się, iż BSS ma tendencję do wybierania bardziej licznych zestawów cech niż FSS (w Tab. 3.1 w 20 przypadkach średnio mniej cech wybrała strategia FSS, w 7 przypadkach mniejszy średni podzbiór należał do BSS, a w jednym przypadku średnie były identyczne). Autor rozprawy przypuszcza, że ma to związek z faktem, iż wewnętrzna wymiarowość problemów jest zazwyczaj niska i FSS ma „krótszą drogę" do znalezienia optimum. Załóżmy, że zbiór cech wybranych przez FSS jest *k*-elementowy i *k* jest małe w stosunku do zadanej wymiarowości. Aby BSS osiągnęła optymalny zestaw, musi podjąć wiele decyzji i prawdopodobnie częściej niż FSS wybrać niewłaściwą ścieżkę. Wynika stąd, że zbiór *k* cech wskazywany przez BSS jest — średnio, w zadaniach o dużej oryginalnej wymiarowości — gorszy niż zbiór *k* cech wybranych przez FSS. Następną konsekwencją jest więc inna liczność optymalnego zbioru cech wskazywanego przez BSS, oznaczmy ją przez *l*. Można przypuszczać, że *l* jest przeważnie większe od *k*, przede wszystkim z tego powodu, iż zmniejszenie *k*-elementowego zbioru złych cech daje małą szansę poprawy estymowanej jakości klasyfikacji, z uwagi na stosunkowo małą przestrzeń poszukiwań.

Sytuacja staje się odwrotna przy przeciwnych założeniach, tj. gdy zadana wymiarowość jest niska. W takim przypadku optymalna liczba cech jest często większa od połowy ogólnej liczby cech, a zatem tym razem strategia FSS ma większe szanse „zbłądzić".

Podobną do opisanej technikę generacji zestawów cech dla klasyfikatora *k*-NN z głosowaniem przedstawiono w pracy (Zenobi i Cunningham, 2001). Algorytm ten różnił się od zaprezentowanego w dwóch aspektach: w bezpośredni sposób kierował się korelacją między klasyfikatorami składowymi oraz stosował zachłanną (ang. *greedy*) metodę generacji zestawów, tj. w każdym kroku wybierał taką modyfikację zestawu cech, która prowadziła lokalnie do najbardziej znaczącej poprawy estymowanej jakości klasyfikacji. Wyniki ze wspomnianej pracy sugerują, iż różnorodność komponentów jest ważniejsza od ich jakości. Wydaje się jednak, że zachłanna metoda uczenia może prowadzić do generacji zbliżonych zestawów cech, a zatem może działać odwrotnie do zamierzeń (minimalizacja korelacji między klasyfikatorami w zespole). Interesującym tematem badań będzie zatem próba połączenia kryterium Zenobi i Cunninghama minimalizującego korelację między klasyfikatorami składowymi z bardziej „swobodną" metodą generacji zestawów cech, np. taką jak przedstawiona w niniejszym podrozdziale.

W najnowszej (2002) pracy Zenobi i Cunningham zaproponowali zespół klasyfikatorów *k*-NN używających różnych zestawów cech z dodatkowym klasyfikatorem *k*-NN działającym na wyjściach komponentów pierwszego etapu. Istnieją przynajmniej dwie możliwości działania klasyfikatora *k*-NN w drugiej fazie: traktowanie wyjść komponentów jako cech symbolicznych (jeśli liczba klas jest większa od dwóch) albo użycie rozmytych wyjść numerycznych.

Rozdział IV
Szybkie szukanie najbliższych sąsiadów metodami heurystycznymi

Jedno z podejść do zagadnienia przyspieszenia klasyfikacji przy użyciu reguły *k*-NN lub jej odmiany obejmuje algorytmy szybkiego szukania najbliższego sąsiada (sąsiadów), tj. zakłada takie wstępne (*off-line*) przetworzenie zbioru, aby później, tj. już w trybie *on-line*, znajdowanie najbliższego sąsiada (sąsiadów) dla zadanej próbki było szybsze od policzenia odległości do wszystkich elementów zbioru. Problem ten jest klasycznym zagadnieniem istotnym lub wręcz kluczowym dla wielu dziedzin informatyki, takich jak drążenie danych (ang. *data mining*), bazy danych (zwłaszcza multimedialne), indeksowanie tekstu, przetwarzanie obrazów i inne. Problem szybkiego szukania najbliższego sąsiada (ang. *nearest neighbor search*, NNS) zwany jest także problemem pocztowym (ang. *post office problem*) (Knuth, 1973), gdyż można go naszkicować w postaci anegdoty: w jaki sposób powinna być przechowywana informacja o lokacjach urzędów pocztowych w danym dużym mieście, aby turysta możliwie najmniejszym wysiłkiem znalazł najbliższą sobie placówkę?

W niniejszym rozdziale omówiono problem szybkiego szukania najbliższego sąsiada z położonym naciskiem na jego liczne i różnorodne zastosowania oraz dokonano przeglądu istniejących heurystyk.

4.1. Definicja problemu i jego wariantów

Formalnie problem NNS można sformułować następująco: mając dany zbiór $P = \{p_1, ..., p_n\}$ w przestrzeni metrycznej X z funkcją odległości df, należy stworzyć struktury danych pozwalające na możliwie szybkie znajdowanie najbliższego sąsiada dla punktów $q \in X$ prezentowanych w trybie *on-line*. Rzecz jasna, podejście „siłowe" (ang. *brute force*) wymaga policzenia odległości z q do wszystkich n punktów ze zbioru P, a zatem koszt szukania najbliższego sąsiada wynosi $O(dn)$, gdzie d jest wymiarem przestrzeni X. Problem został po raz pierwszy postawiony w końcu lat 60-tych (Minsky i Papert, 1969) i mimo ponad 30 lat intensywnych badań nadal jesteśmy dalecy od satysfakcjonujących rozwiązań, zwłaszcza w wysokich wymiarach. Dla $d = 2$

zastosowanie diagramu Voronoi'a prowadzi do optymalnego rozwiązania z przetwarzaniem wstępnym w czasie $O(n \log n)$, kosztem pamięciowym $O(n)$ i czasem szukania $O(\log n)$ dla pojedynczej próbki (Preparata i Shamos, 1985). Już dla $d = 3$ nie są znane algorytmy z czasem przetwarzania wstępnego zbliżonym do liniowego w *n*, a jednocześnie z czasem szukania zbliżonym do logarytmicznego w *n*.

Druga, bardziej praktyczna trudność, polega na małej znajomości istniejących algorytmów nie tylko wśród programistów, ale nawet wśród naukowców zajmujących się rozpoznawaniem obrazów i dziedzinami pokrewnymi: wiąże się to z faktem, iż większość podawanych w literaturze propozycji to algorytmy dość skomplikowane i trudne w implementacji.

Problem szukania najbliższego sąsiada niejednokrotnie rozważany był w przestrzeni R^d, ale w niektórych zastosowaniach można tę przestrzeń zawęzić np. do *d*-wymiarowej kostki Hamminga, tj. zbioru wszystkich punktów postaci $\{0,1\}^d$. O ile w algorytmach z dowiedzionym subliniowym czasem szukania stosowano zwykle metrykę euklidesową, o tyle rozwiązania heurystyczne często działają w dowolnych przestrzeniach metrycznych (obszerny przegląd tych ostatnich zawiera praca (Chávez i in., 2001)).

A oto kilka przykładowych konkretnych zagadnień, w których rzeczą istotną jest szybkie znalezienie najbliższych — lub możliwie bliskich — sąsiadów zadanego obiektu:

- wyszukiwanie możliwie bliskiego wzorca w bazach multimedialnych (identyfikacja twarzy, głosu, odcisków palców, wykrywanie plagiatów muzycznych *etc*.);
- szukanie w indeksowanym tekście; wygodną funkcją jest dopuszczanie pewnej liczby różnic pomiędzy szukanym a testowanym wzorcem (najczęściej są to literówki — praktycznie nie do uniknięcia w bardzo dużych bazach);
- grupowanie dokumentów WWW o pokrewnej treści, np. na potrzeby wyszukiwarek sieciowych;
- przybliżone dopasowywanie fragmentów obrazów w kompresji video;
- predykcja zachowań na giełdzie;
- przybliżone (z powodu możliwych mutacji) szukanie sekwencji proteinowych lub DNA;
- typowe przykłady klasyfikacji w rozpoznawaniu obrazów (m. in. analiza zdjęć medycznych, przemysłowych, kontrola jakości, *remote sensing*).

Mimo iż często argumentowano, że istniejące algorytmy szybkiego szukania najbliższego sąsiada, przynajmniej w wersji dokładnej, nie przydają się, gdy wymiar problemu przekracza ok. 10–15, to dokładniejsza analiza — zarówno teoretyczna, jak i eksperymentalna — pokazała, że istotniejsza od „zadanej" wymiarowości jest wymiarowość wewnętrzna (ang. *intrinsic dimensionality*), której dobrej wyznacznikiem jest wymiar fraktalny Hausdorffa danego problemu (Pagel i in., 2000). Eksperymenty w przytoczonej pracy, gdzie techniką heurystyczną szukania najbliższego sąsiada były R*-drzewa (p. Rozdz. 4.2), wykazały, że nawet na zbiorach o wymiarze 100 algorytmy szybkiego szukania najbliższego sąsiada mogą się dobrze sprawdzać, o ile wymiar Hausdorffa jest odpowiednio niski (nie przekraczał 6 w cytowanych eksperymentach).

Pomimo tej „pocieszającej" wiadomości należy stwierdzić, że częste są sytuacje, w których istniejące algorytmy dokładnego (ang. *exact*) szukania najbliższego sąsiada (NS) zawodzą. W pracy (Beyer i in., 1999) pojawia się sugestia, iż w bardzo wysokich wymiarach najlepszą metodą szukania NS jest na ogół pełny sekwencyjny przegląd; sekwencyjność jest istotna przy przeszukiwaniu ogromnych baz, które nie mieszczą się w pamięci głównej, i gdzie korzystanie z indeksu powoduje „losowy" dostęp do danych na dysku, co negatywnie wpływa na koszt operacji wejścia/wyjścia. Zrozumiałe jest zatem, iż rozważane zagadnienie doczekało się również wersji przybliżonej, o złagodzonych wymaganiach (A-NNS – *approximate nearest neighbor search*). W wariancie A-NNS rozwiązanie uważa się za znalezione, jeśli potrafimy wskazać element p_i taki, że

$$d(q, p_i) \le \alpha \cdot d(q, p_j), \tag{4.1}$$

gdzie p_j jest „prawdziwym" najbliższym sąsiadem q, zaś $\alpha \ge 1$ jest współczynnikiem decydującym o jakości rozwiązania (im większe α, tym łatwiej otrzymać rozwiązanie, ale i jest ono na ogół gorsze; $\alpha = 1$ sprowadza A-NNS do klasycznej wersji NNS). Dla tej wersji problemu w ostatnich latach osiągnięto interesujące wyniki (Kushilevitz i in., 1998; Indyk i Motwani, 1998).

Za stosowaniem wersji A-NNS może też przemawiać wyliczenie z monografii (Fukunaga, 1990), w myśl którego stosunek przeciętnej odległości do (k+1)-szego sąsiada do przeciętnej odległości do k-tego sąsiada w kostce jednostkowej przy rozkładzie jednostajnym wynosi

$$\frac{E\{d_{(k+1)NN}\}}{E\{d_{kNN}\}} \approx 1 + \frac{1}{kd}, \tag{4.2}$$

a zatem maleje wraz z rosnącym wymiarem *d*.

Proponowano także algorytmy dla sytuacji, gdy rozkład wejściowych próbek jest znany (Yianilos, 1993; Clarkson, 1999).

W dalszych rozważaniach zadany obiekt będzie oznaczany przez *q* (od ang. *query* – zapytanie).

W praktycznych zastosowaniach spotyka się najczęściej trzy rodzaje zapytań wiążących się z omawianym problemem:

(a) znaleźć wszystkie obiekty leżące względem *q* w odległości nie przekraczającej zadanego *r*;

(b) znaleźć najbliższego sąsiada *q*; niekiedy zadaje się również maksymalną odległość *r* do najbliższego sąsiada — jeśli leży on dalej, to wynikiem poszukiwań jest zbiór pusty;

(c) znaleźć *k* najbliższych sąsiadów *q*.

4.2. Przegląd istniejących heurystyk

Większość istniejących metod szukania najbliższego sąsiada to heurystyki. Mimo iż koszta pamięciowo-czasowe przetwarzania wstępnego są zazwyczaj do zaakceptowania, należy pamiętać, że efektywność tych rozwiązań maleje wraz z rosnącym wymiarem (zwłaszcza, jak już zostało powiedziane, z wymiarem wewnętrznym problemu). Szeroko przyjęta jest hipoteza, iż dla dowolnego algorytmu złożoność albo wstępnej obróbki (koszt pamięciowy i czasowy), albo szukania najbliższego sąsiada rośnie wykładniczo z wymiarem. Przypuszczenie to potocznie określa się mianem „przekleństwa wymiarowości" (ang. *curse of dimensionality*). Mimo iż wciąż brakuje formalnego dowodu hipotezy, podawano pewne argumenty przemawiające za jej przyjęciem (Borodin i in., 1999).

Poniżej dokonano krótkiego przeglądu rozwiązań heurystycznych dla rozważanego problemu, poczynając od algorytmów i struktur danych działających w dowolnych przestrzeniach metrycznych.

Najstarszą strukturą danych służącą do wyszukiwań w przestrzeniach metrycznych jest *drzewo Burkharda–Kellera* (ang. *Burkhard–Keller tree*, *BK-tree*) (Burkhard i Keller, 1973). Oryginalnie zostało ono zaprojektowane pod kątem metryk dyskretnych, ale można je zaadaptować również do metryk ciągłych. W drzewie Burkharda–

Kellera korzeniem jest dowolnie wybrany element *a*, który posiada pewną liczbę potomków. W potomku *i*-tym buduje się rekursywnie drzewo dla wszystkich elementów *P* leżących w odległości *i* od *a*. Rekursja postępuje aż do otrzymania pojedynczego elementu albo co najwyżej *b* elementów (może też być ograniczana maksymalną wysokością drzewa *h*). Stosowane są dwa rodzaje zapytań. Zapytania typu (a) obsługiwane są następująco: zaczynamy od korzenia i przechodzimy przez wszystkich potomków i takich, że $d(a,q)-r \le i \le d(a,q)+r$; proces postępuje rekursywnie. Jeśli dojdziemy do liścia, liczymy po kolei odległości do wszystkich przechowywanych tam elementów. Zapytania typu (b) realizowane są w podobny sposób, różnica zasadniczo polega na tym, że tym razem odpowiednikiem stałego *r* w zapytaniu typu (a) jest stale zmieniająca się (malejąca) odległość do aktualnie najbliższego sąsiada. Lista potomków, których należy odwiedzić w czasie szukania, stopniowo zatem zawęża się.

W tej samej pracy zaproponowano rekursywny podział zbioru *P* na rozłączne podzbiory P_i i wybór reprezentanta a_i dla każdego podzbioru. Dla każdego P_i liczymy odległość $k_i = \max\{d(a_i, x) : x \in P_i\}$. W czasie szukania najbliższego sąsiada liczymy odległości od *q* do każdego a_i rozpatrując następnie odpowiednie zbiory w kolejności od najmniejszej do największej odległości. Wartości k_i pozwalają ustalić, w których zbiorach P_i nie ma szans na znalezienie najbliższego sąsiada. Nietrudno zauważyć, że wydajność struktury musi zależeć od wyboru podzbiorów P_i i ich reprezentantów a_i. Autorzy zaproponowali złożone kryterium pozwalające dokonać wyboru tych parametrów. W praktyce jednak algorytm niewiele odbiega od metody z drzewem B–K.

Jednym z najpopularniejszych technik jest podejście wykorzystujące strukturę zwaną *drzewem awantażowym* lub *vp-drzewem* (ang. *vantage-point tree*,[9] *vp-tree*) (Uhlmann, 1991; Yianilos, 1993). Jest to drzewo zrównoważone, którego korzeniem jest dowolny element *a*, lewe poddrzewo tworzą wszystkie elementy w odległości od *a* nie przekraczającej mediany zbioru wszystkich odległości od *a*, zaś prawe poddrzewo tworzą elementy w odległości od *a* nie mniejszej niż owa odległość medianowa. (Należy zwrócić uwagę, że niektóre elementy mogą pojawić się w obu poddrzewach.) Oznaczmy opisaną medianę przez *m*. Przy zapytaniu typu (a) liczymy odległość $d = d(a,q)$. Jeśli $d - r < m$, to wkraczamy do lewego poddrzewa, jeśli zaś $d + r \ge m$, to do prawego (możliwe jest wejście do obu poddrzew).

[9] Uhlmann używa terminu „metric tree” (Uhlmann, 1991).

Rozważano także drzewa awantażowe o podstawie *m* (Brin, 1995). W takim przypadku zbiór odległości dzieli się na *m* równolicznych przedziałów (zamiast podziału medianą na dwie części w oryginalnym pomyśle).

Inną propozycją jest *gh-drzewo* (ang. *generalized-hyperplane tree*, *gh-tree*) (Uhlmann, 1991). Dla każdego węzła wybierane są dwa elementy, które wyznaczają hiperpłaszczyznę rozdzielającą elementy lewego i prawego poddrzewa. Przemieszczanie się po drzewie jest analogiczne do przypadku drzewa awantażowego. W wyższych wymiarach gh-drzewo może być bardziej efektywne od vp-drzewa. Idea partycjonowania przestrzeni przy użyciu dwóch punktów pojawiła się już w pracy (Kalantari i McDonald, 1983), gdzie na obu elementach wyznaczających płaszczyznę była rozpinana kula o promieniu równym odległości do najdalszego punktu danej strony płaszczyzny (tj. z danego poddrzewa). Znajomość promienia pozwala na eliminację danego poddrzewa przy zapytaniach typu (a).

Uogólnieniem gh-drzewa jest struktura GNAT (ang. *Geometric Near-neighbor Access Tree*) (Brin, 1995), będąca drzewem o podstawie *m* i wykorzystująca drugą z opisanych wyżej idei z pracy (Burkhard i Keller, 1973).

Warto ponadto wspomnieć nowsze struktury dynamiczne, jak *M-drzewo* (ang. *metric tree*, *M-tree*) (Ciaccia i in., 1997) oraz *„drzewo odchudzone”* (ang. *Slim-tree*) (Traina Jr. i in., 2000), które budowane są od dołu do góry (w przeciwieństwie do większości innych podejść). *Slim-tree* stara się mimimalizować stopień nakładania się podzbiorów zbioru *P* umieszczonych w węzłach drzewa, m. in. przez zastosowanie heurystyki opartej na konstrukcji minimalnego drzewa rozpinającego (ang. *Minimal Spanning Tree*, MST) (Kruskal, 1956) do rozszczepiania węzłów.

Węższą klasę przestrzeni niż przestrzenie metryczne stanowią przestrzenie wektorowe, w których obiekty są uporządkowanymi krotkami liczb rzeczywistych, co pozwala na wykorzystanie geometrycznych właściwości zbioru, np. poprzez rzutowanie. Prawdopodobnie pierwszym algorytmem z tej klasy był algorytm szukania korzystający z *drzewa k-wymiarowego* (ang. *k-dimensional tree*, *kd-tree*) (Bentley, 1975; Friedman i in., 1977; Bentley, 1979). Konstrukcja drzewa *k*-wymiarowego polega na rekursywnym rozcinaniu podzbiorów zbioru odniesienia pojedynczymi płaszczyznami równoległymi do jednej z osi, w taki sposób, aby oba powstałe w danym kroku poddrzewa zawierały w przybliżeniu taką samą liczbę obiektów. W każdym kroku wybierana jest ta z *d* osi współrzędnych, która maksymalizuje dyspersję dla bieżącego podzbioru (dyskusję metod pomiaru dyspersji zawiera praca (Friedman i in., 1977)).

Liście drzewa mogą zawierać więcej niż jeden obiekt. Szukanie polega na wyznaczeniu liścia, w którym znalazłaby się próbka *q*, następnie znalezieniu najbliższego sąsiada *q* wśród próbek zawartych w liściu, i wreszcie ewentualnym przeszukaniu wszystkich innych liści (poczynając od sąsiednich), w których może znaleźć się najbliższy sąsiad *q*. Rzecz jasna, promień szukania w trakcie tego procesu może się stopniowo zmniejszać, ograniczając zbiór liści, które należy poddać inspekcji.

R-drzewa (ang. *R-trees*) (Guttman, 1984) grupują próbki zbioru odniesienia w dobieranych do nich hiperprostokątach. W czasie szukania zbiór hiperprostokątów, które mogą zawierać najbliższego sąsiada *q* stopniowo się zmniejsza. Hiperprostokąty na tym samym poziomie drzewa mogą zachodzić na siebie (ang. *overlap*), co przy znaczącym stopniu nakładania może prowadzić do niskiej prędkości szukania, ale rozwiązanie takie zapewnia pamięciową oszczędność struktury (brak pustych liści). Proponowano kilka modyfikacji R-drzewa, różniących się m. in. kryterium rozszczepiania węzła (głównym celem jest minimalizacja nakładania się hiperprostokątów na tym samym poziomie), z których najbardziej udaną wersją było *R*-drzewo* (Beckmann i in., 1990).

Strukturami silnie inspirowanymi R-drzewem były *SS-drzewo* (White i Jain, 1995) i *SR-drzewo* (Katayama i Satoh, 1997); w pierwszym przypadku obiekty w węzłach są ograniczone sferami (w odróżnieniu od prostokątów w R-drzewie), zaś w drugim zarówno sferami, jak i prostokątami.

Dość oryginalną koncepcję zaprezentowano w (Weber i in., 1998). Autorzy dowodzą, że przy pewnych założeniach dotyczących rozkładu próbek i przy odpowiednio wysokiej wymiarowości, żadna struktura indeksująca nie pozwoli na efektywne znajdowanie najbliższych sąsiadów. Autorzy proponują zatem zaprzestanie tej „przegranej już wojny", natomiast skupienie się na poprawieniu (o stały czynnik) szybkości sekwencyjnego przeglądu. Konkretnie, proponują oni, przy pomocy wprowadzonej struktury zwanej „VA-file" (VA – *Vector Approximation*), szukanie najbliższego sąsiada w zbiorze próbek o skwantyzowanych wartościach cech (czyli poddanych kompresji stratnej). Idea polega na szybszym testowaniu zredukowanych w ten sposób danych przy możliwie rzadkiej konieczności odwołania się do oryginalnych danych. W praktyce daje to często przyspieszenie 2–5-krotne względem liniowego przeglądu na oryginalnych danych przy wymiarowości bazy ok. 10–15. Praca (Berchtold i in., 2000) z kolei łączy podejście klasyczne (tj. z indeksem) z kwantyzacją danych, oferując w wysokich wymiarach szybkość wyszukiwania do 3 razy większą niż *VA-file*. Jeszcze jednym

podejściem inspirowanym strukturą *VA-file* jest algorytm „Active Vertice" (Balko i Schmitt, 2002), gdzie stopień kwantyzacji danych dobierany jest lokalnie. Autorzy argumentują, że przy tym samym stopniu kompresji (tj. średniej kwantyzacji) zbioru odniesienia zaproponowane podejście niesie więcej informacji niż struktura *VA-file*, tj. wymaga mniejszej liczby odwołań do oryginalnych danych.

Interesująca koncepcja przyspieszenia klasyfikacji 1-NN i *k*-NN została przedstawiona w pracach (Skubalska-Rafajłowicz i Krzyżak, 1995; 1996). Wielowymiarowa przestrzeń jest zindeksowana krzywą wypełniającą przestrzeń. Rozpatrywano uogólnioną krzywą Sierpińskiego i Peano. Szukanie *k* najbliższych sąsiadów sprowadza się do wyznaczenia położenia zadanej próbki na danej krzywej (czyli w przestrzeni jednowymiarowej) wypełniającej *d*-wymiarową przestrzeń, a następnie znalezienia sąsiadów metodą wyszukiwania binarnego. Koszt klasyfikacji jednego obiektu jest zatem praktycznie logarytmiczny w *n*. Niestety, rozrzut wyników dla różnych kombinacji parametrów (wybrana krzywa oraz metoda obejścia przestrzeni) i różnych zbiorów w porównaniu z oryginalną *k*-NN raczej zniechęca do stosowania metody w przypadkach, gdy głównym celem jest dostatecznie wysoka jakość klasyfikacji.

Ulepszenia idei Skubalskiej-Rafajłowicz i Krzyżaka (dokonane prawdopodobnie niezależnie od pierwowzoru) zaprezentowane są w pracach Shepherda i in. (1999) oraz Liao i in. (2001), w których użyto więcej niż jednej listy próbek ułożonych wzdłuż krzywej Hilberta, przy czym wielość list otrzymywana jest przez translacje zbioru odniesienia (w praktyce, dla wysokich wymiarów, najczęściej potrzeba aż kilkudziesięciu takich list). W pierwszej z wymienionych prac transformacje zbioru są *de facto* losowe; algorytm Liao i in. natomiast używa ustalonych wektorów translacji i daje wyniki, o których wiadomo, że odbiegają od rozwiązania optymalnego (tj. dokładnego najbliższego sąsiada) o co najwyżej zadany czynnik, a zatem znaleziony przybliżony najbliższy sąsiad spełnia definicję określoną wzorem (4.1). Jeszcze inną propozycją z tej rodziny jest użycie krzywej wypełniającej przestrzeń jako pierwszego etapu szukania: znaleziony na tej krzywej najbliższy sąsiad próbki *q* wskaże poddrzewo, do którego elementów należy dodatkowo policzyć odległości, aby z dużym prawdopodobieństwem wskazać globalnie najbliższego sąsiada *q* (Tan i in., 1999). Konkretna krzywa dla danego zadania (celem jest minimalizacja wskazywanego poddrzewa przy jednoczesnej wysokiej jakości wskazywanego sąsiada) jest znajdowana przy użyciu algorytmów genetycznych.

Bardzo prostą heurystykę szukania najbliższego sąsiada, polegającą na odrzucaniu wielu odległych próbek w czasie krótszym od wyznaczenia do nich odległości, stosował autor rozprawy w pracy (Grabowski, 1999); idea wykorzystywana jest praktycznie we wszystkich schematach przeszukujących bazę danych przy dowolnej przestrzeni metrycznej, począwszy od drzewa Burkharda–Kellera. Na początku procedury należy obliczyć i zapamiętać odległości wszystkich obiektów od jednego ustalonego obiektu *A* (np. pierwszego w zbiorze). Jeżeli teraz dla pewnego obiektu *B* dotychczas znalezionym najbliższym sąsiadem jest *C*, to przed obliczeniem odległości od *B* do aktualnego obiektu *D*, należy sprawdzić, czy zachodzi nierówność:

$$\left|d(D,A)-d(B,A)\right| > d(B,C). \tag{4.3}$$

Jeżeli nierówność (4.3) ma miejsce, to próbkę *D* można odrzucić bez liczenia do niej odległości. Wynika to wprost z warunku nierówności trójkąta. Rzecz jasna, wszystkie występujące w nierówności (4.3) odległości zostały policzone już wcześniej, a więc sprawdzenie tego warunku może się opłacać, gdy spełniony jest on z odpowiednią częstością, a liczba cech jest stosunkowo duża (gdyż wówczas wyznaczanie odległości jest kosztowne). Warto zauważyć, że proponowana heurystyka jest bardziej opłacalna przy metryce miejskiej niż przy metryce euklidesowej. Wynika to z faktu, iż dość kosztowne liczenie pierwiastka w tradycyjnej implementacji można pominąć, natomiast nie dałoby się tego zrobić przy korzystaniu z nierówności (4.3). W praktyce, opisanych warunków (kryteriów negatywnych) może być kilka — obiekt, który nie został odrzucony za pierwszym razem, może zostać odrzucony nie spełniając kolejnego kryterium.

Rozdział V
Algorytm szybkiego deterministycznego szukania najbliższego sąsiada w metryce miejskiej

W rozdziale dokonano krótkiego przeglądu algorytmów z gwarantowanym dla najgorszego lub średniego przypadku subliniowym w liczności zbioru czasem szukania, a następnie zaproponowano nowy algorytm deterministycznego szukania najbliższego sąsiada w metryce miejskiej (Grabowski, 2001a, Grabowski i Baranowski, 2002). Algorytm ten cechuje się elastycznością (możliwe jest ustawienie współczynnika kompromisu między czasem szukania a kosztem pamięciowo-czasowym wstępnej obróbki) oraz znaczną prostotą koncepcji. Ponadto przedstawiono wyniki działania implementacji algorytmu na zbiorach sztucznych oraz na zbiorze Iris. Wybór metryki miejskiej został uzasadniony korzystnymi, w stosunku np. do metryki euklidesowej, wartościami miary Cháveza–Navarro (2001) dla szeregu rozpatrywanych zbiorów naturalnych i sztucznych.

5.1. Algorytmy z subliniowym czasem szukania

Liczba metod szukania NS, dla których udowodniono, choćby tylko w przypadku średnim, subliniowość czasu szukania w liczności zbioru n, jest stosunkowo mała. Pierwszy algorytm, działający w R^d (Dobkin i Lipton, 1976), szukał NS w czasie o złożoności $O\left(2^d \log n\right)$, jednak koszt wstępnej obróbki[10] wynosił aż $O\left(n^{2^{d+1}}\right)$. Zastosowana metoda wykorzystywała diagram Voronoi'a i obniżała rekursywnie wymiarowość problemu. Clarkson (1988) zredukował koszt wstępnego przetwarzania do $O\left(n^{\lceil d/2 \rceil (1+\delta)}\right)$ (wartość oczekiwana), ale czas szukania wzrósł do $O\left(2^{O(d \log d)} \log n\right)$. W późniejszych propozycjach (Yao i Yao, 1985; Agarwal i Matoušek, 1992; Matoušek, 1992) również czas szukania rośnie wykładniczo w d. W bardziej teoretycznej wersji algorytmu Clarksona (Meiser, 1993) czas szukania wygląda interesująco: $O\left(d^5 \log n\right)$, ale koszt wstępnej obróbki wynosi $O\left(n^{d+\delta}\right)$.

[10] W tej klasie algorytmów zwykle nie rozróżnia się pomiędzy kosztem pamięciowym a czasowym wstępnej obróbki. Są one przeważnie zbliżone.

Należy podkreślić, że nie wszystkie z wymienionych algorytmów gwarantują subliniowy czas szukania w najgorszym przypadku, np. algorytmy Clarksona i Meisera są probabilistyczne. Prezentowany w niniejszej pracy algorytm jest natomiast algorytmem deterministycznym, tj. gwarantuje subliniowość także w najgorszym przypadku.

5.2. Zalety metryki miejskiej

W pracy (Brin, 1995) pojawiła się sugestia, że histogram odległości między punktami zbioru odniesienia *P* jest ściśle związany z wewnętrzną wymiarowością przestrzeni metrycznej. Histogram bardziej scentrowany odpowiada przestrzeni o wyższej wewnętrznej wymiarowości (a zatem „trudniejszej" dla algorytmów szybkiego szukania sąsiadów). W pracy (Chávez i Navarro, 2001) autorzy wykorzystali intuicję Brina i wprowadzili miarę trudności wyszukiwania najbliższych sąsiadów w danym zbiorze, odnoszącą się do dowolnych przestrzeni metrycznych. Pokazano, że dogodną miarą takiej trudności — odnoszącą się dla dowolnych algorytmów szukania NS bazujących na *k* niezależnych punktach odniesienia (ang. *pivots*) — jest iloraz $\mu^2/2\sigma^2$, gdzie μ jest średnią odległością między losową parą próbek w zbiorze, zaś σ odchyleniem standardowym histogramu odległości między wszystkimi parami próbek w zbiorze.

W niniejszym podrozdziale policzono wartości miary Cháveza–Navarro dla szeregu zbiorów rzeczywistych i sztucznych oraz metryk: miejskiej i euklidesowej.

Tab. 5.1 przedstawia wartości μ, σ oraz $\mu^2/2\sigma^2$ policzone dla następujących zbiorów: Ferrites, Remotes, pięciu zbiorów UCI oraz zbiorów losowych próbek o rozkładzie jednostajnym w hiperkwadracie jednostkowym o wymiarowości od 2 do 8 (Random2d, ..., Random8d). Wszystkie rzeczywiste zbiory zostały znormalizowane.

Zgodnie z oczekiwaniami, średnia odległość między parą punktów w metryce miejskiej jest znacznie większa niż w metryce euklidesowej, ale jeszcze większa jest dysproporcja między odchyleniami standardowymi odległości dla tych metryk. Warto zauważyć, że dysproporcja między wartościami miary trudności wyrażonej ilorazem $\mu^2/2\sigma^2$ rośnie wraz z wymiarem dla zbiorów o jednostajnym rozkładzie próbek (Random2d, ..., Random8d). Większa wartość rozważanej miary sugeruje oczywiście większą trudność w zaprojektowaniu algorytmu szukania NS w metryce euklidesowej; można także przypuszczać, że klasyfikacja, czyli *de facto* dyskryminacja klas w przestrzeni, jest trudniejsza w metryce euklidesowej, która — mówiąc potocznie —

„spłaszcza" odległości. Rzecz jasna, rozumowanie nie jest ścisłe, gdyż wywody Chávеza i Navarro odnosiły się tylko do pewnej klasy algorytmów (w dodatku przy pewnych upraszczających założeniach), jednak również z doświadczeń autora rozprawy wynika, że jakość klasyfikacji jest zazwyczaj nieco lepsza w metryce miejskiej[11] i w świetle powyższych eksperymentów można przypuszczać, iż nie jest to przypadek. Ponadto metryka miejska jest nieco mniej kosztowna obliczeniowo. Z tych powodów autor rozprawy preferuje tę metrykę i w niej działa przedstawiany algorytm.

Tabela 5.1: Porównanie parametrów rozkładu odległości
w zbiorach rzeczywistych i syntetycznych przy metryce miejskiej oraz euklidesowej

zbiór	liczba cech	liczność zbioru	metryka miejska			metryka euklidesowa		
			μ	σ	$\mu^2/2\sigma^2$	μ	σ	$\mu^2/2\sigma^2$
Ferrites	30	5903	19,68	17,19	**0,66**	5,93	4,75	**0,78**
Remotes	9	5124	10,16	4,04	**3,17**	3,99	1,48	**3,64**
Bupa	6	345	6,00	3,26	**1,69**	3,06	1,62	**1,78**
Glass	9	214	8,30	5,19	**1,28**	3,65	2,17	**1,42**
Iris	4	150	4,52	2,57	**1,55**	2,50	1,32	**1,80**
Pima	8	768	8,40	3,13	**3,60**	3,76	1,37	**3,76**
Wine	13	178	14,59	4,69	**4,83**	4,89	1,44	**5,80**
Random2d	2	3000	0,67	0,33	**2,02**	0,53	0,25	**2,24**
Random3d	3	3000	1,01	0,41	**2,99**	0,67	0,25	**3,53**
Random4d	4	3000	1,33	0,47	**4,02**	0,78	0,25	**4,90**
Random5d	5	3000	1,66	0,52	**5,03**	0,87	0,25	**6,28**
Random6d	6	3000	2,00	0,58	**6,04**	0,97	0,25	**7,71**
Random7d	7	3000	2,34	0,63	**6,98**	1,05	0,25	**9,04**
Random8d	8	3000	2,67	0,66	**8,10**	1,13	0,25	**10,66**

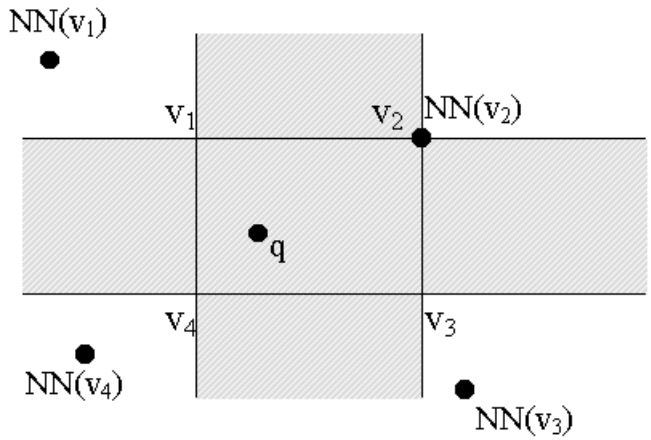
5.3. Proponowany algorytm szukania NS w metryce miejskiej

W niniejszej sekcji zaprezentowano algorytm szybkiego deterministycznego szukania najbliższego sąsiada w metryce miejskiej. Algorytm został przedstawiony po raz pierwszy w (Grabowski, 2001a), zaś wyniki testów implementacji oraz szerszą analizę poprawności metody podano w (Grabowski i Baranowski, 2002). Czas szukania wynosi $O\left(d\log\left(\frac{n}{k}\right)+d\cdot 2^d+d^2\cdot k\right)$, pamięć potrzebna dla przetwarzania wstępnego to $O\left(\left(\frac{n}{k}\right)^d\right)$, zaś czas wstępnego przetwarzania wynosi $O\left(\left(\frac{n}{k}\right)^d\cdot n\cdot d\right)$. Współczynnik k jest kompromisem między kosztem przetwarzania wstępnego a czasem szukania.

Opis algorytmu podano na początku dla wersji najszybszej, tj. z $k=1$. Przed przeczytaniem formalnego opisu warto być może spojrzeć na Rys. 5.1, który demonstruje ideę algorytmu na przykładzie dwuwymiarowym. Jedna z próbek NN(v_1), ...,

[11] Podobną obserwację potwierdza dr A. Jóźwik (korespondencja prywatna, 2000–2002).

NN(v_4) musi być najbliższym sąsiadem q. W wierzchołku v_2 znajduje się (przypadkowo) próbka ze zbioru P (która oczywiście jest najbliższym sąsiadem dla tego wierzchołka).



Rys. 5.1. Idea algorytmu szybkiego szukania najbliższego sąsiada w metryce miejskiej

Przetwarzanie wstępne

Wstępna obróbka zasadniczo polega na podzieleniu d-wymiarowej przestrzeni X na sieć hiperprostokątów (zwanych danej „komórkami") o ścianach równoległych do osi układu współrzędnych. Poniżej podano formalny opis tego procesu.

Niech L_j, $j = 1..d$, będzie posortowaną listą j-tych współrzędnych wszystkich punktów z P, gdzie $P = \{p_1, ..., p_n\}$. Dokładniej $L_j = \left[p_0^j, p_1^j, p_2^j, \ldots, p_n^j, p_{n+1}^j\right]$, gdzie p_0^j to –INF, a p_{n+1}^j to +INF (ze względu na spójność opisu algorytmu, plus/minus nieskończoność traktowane są jako liczby), zaś p_i^j, $i = 1..n$, to kolejne posortowane j-te współrzędne punktów z P. Komórką nazywamy iloczyn kartezjański $\left[p_{i_1}^1, p_{i_1+1}^1\right] \times \left[p_{i_2}^2, p_{i_2+1}^2\right] \times \ldots \times \left[p_{i_d}^d, p_{i_d+1}^d\right]$, $0 \le i_k \le n$, $k = 1..d$. Innymi słowy, komórki powstają w wyniku przeprowadzenia przez każdy punkt ze zbioru P d hiperpowierzchni (d–1)-wymiarowych, z których każda jest prostopadła do innej osi układu współrzędnych.

Dla każdego wierzchołka każdej komórki (z wyjątkiem wierzchołków o wartości +/–INF na przynajmniej jednej współrzędnej) znajdowany jest i zapamiętywany najbliższy sąsiad z P. Wstępna obróbka jest zakończona.

Szukanie

Szukanie najbliższego sąsiada jest bardzo proste. Na początku znajdowana jest komórka zawierająca zadaną próbkę q (do ustalenia indeksu komórki stosuje się d-krotnie wyszukiwanie binarne na listach L_j, $j = 1..d$). Jeden z najbliższych sąsiadów dla wierzchołków tej komórki musi być najbliższym sąsiadem próbki q. Uzasadnienie tego faktu znajduje się w następnej podsekcji.

O poprawności algorytmu

Rozważmy alternatywną (ale oczywiście również dającą poprawne wyniki) wersję przetwarzania wstępnego. Podział przestrzeni na sieć hiperprostokątów pozostaje ten sam, różnica polega zaś na znajdowaniu dla wierzchołków „siatki" większej liczby inaczej niż poprzednio określonych sąsiadów. Do opisu potrzebne są dodatkowe oznaczenia.

Niech $X_{[b_1,b_2,\ldots,b_d]}(p)$, gdzie $[b_1, b_2, \ldots, b_d]$ jest wektorem binarnym, zaś p dowolnym punktem z X, będzie zbiorem wszystkich punktów $s \in X$, takich że $s^j \geq p^j$, gdy $b_j = 1$ i $s^j \leq p^j$, gdy $b_j = 0$, $j = 1..d$. Każdy punkt $p \in X$ umożliwia dekompozycję przestrzeni X na 2^d podzbiorów, takich że każdemu podzbiorowi odpowiada jeden d-wymiarowy wektor binarny (nie ma znaczenia fakt, iż zbiory te nie są rozłączne, tj. punkty leżące na ścianach komórek należą do dwóch lub więcej opisanych podzbiorów przestrzeni).

W szczególności, każdy wierzchołek każdej komórki (z wyjątkiem wierzchołków o przynajmniej jednej składowej równej +/–INF) umożliwia dekompozycję przestrzeni X na 2^d części. Niech C będzie komórką, zaś $v_i(C)$ dowolnym wierzchołkiem komórki C takim, że żadna współrzędna tego wierzchołka nie jest równa +/–INF. Niech q będzie dowolnym punktem z C, zaś $B = [b_1, b_2, \ldots, b_d]$ będzie wektorem binarnym, takim iż $q \in X_B(v_i(C))$. Przyjmujemy oznaczenie, iż $-B$ oznacza wektor powstały z B przez zanegowanie wszystkich jego binarnych składowych. Wśród punktów należących do P znajdujemy taki punkt $p_k \in X_{-B}(v_i(C))$, że p_k jest najbliższym sąsiadem $v_i(C)$ w $X_{-B}(v_i(C))$ w sensie metryki miejskiej. Innymi słowy, znajdowany jest najbliższy sąsiad $v_i(C)$ w „ćwiartce" przestrzeni leżącej „po przekątnej". Tacy sąsiedzi są znajdowani i zapamiętywani dla każdego wierzchołka każdej komórki (z wyjątkiem wierzchołków o wartości +/–INF na przynajmniej jednej współrzędnej). Na tym

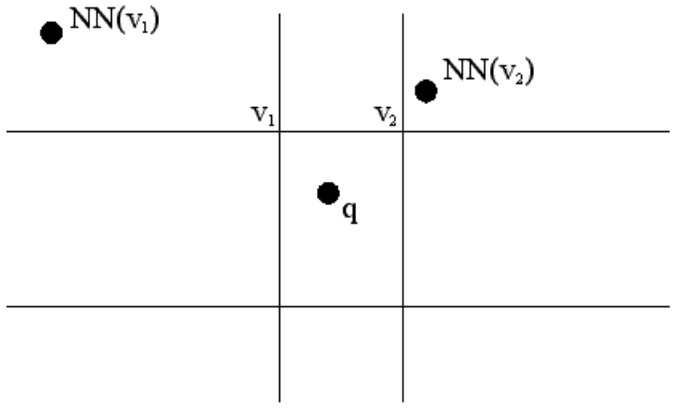
kończy się wstępna obróbka alternatywnego algorytmu. Szukanie jest analogiczne po poprzedniej wersji: dla komórki zawierającej zadaną próbkę testową q odczytywani są najbliżsi sąsiedzi w „ćwiartkach" leżących „po przekątnej" i jeden z tych sąsiadów (tj. najbliższy z nich) jest najbliższym sąsiadem próbki q.

Poprawność opisanej procedury wynika z poniższej własności metryki miejskiej: dla dowolnych punktów A, B, C

$$A_i \le B_i \le C_i \Rightarrow d(A_i, C_i) = d(A_i, B_i) + d(B_i, C_i), \quad i = 1 \ldots d. \tag{5.1}$$

Idea algorytmu została ukazana na Rys. 5.1 (przykład 2-wymiarowy). Jak zostało powiedziane, jedna z próbek NN(v_1), ..., NN(v_4) musi być najbliższym sąsiadem q, gdyż wynika to z faktu, że zacienione pasy z definicji algorytmu nie mogą zawierać żadnej próbki z P.

Wracamy teraz do oryginalnego algorytmu. Jest oczywiste, że „globalnie" najbliższy sąsiad dla danego wierzchołka nie musi należeć do „ćwiartki" opisanej w alternatywnej wersji algorytmu. Przykładowo na Rys. 5.2 najbliższym sąsiadem dla v_1 jest NN(v_2), który leży bliżej niż jego najbliższy sąsiad „z ćwiartki", NN(v_1). Łatwo jednak zauważyć, że jeśli NN(v_2) jest najbliższym sąsiadem dla v_1, to w metryce miejskiej musi być także globalnie najbliższym sąsiadem dla v_2. W obu przedstawionych wersjach algorytmu listy sąsiadów związane z daną komórką w obu wersjach algorytmu bedą zawierały najbliższego sąsiada dla danej próbki z tej komórki.



Rys. 5.2. Przykład sytuacji, w której najbliższy sąsiad pewnego wierzchołka (v_1) należy do innej „ćwiartki" przestrzeni niż dany wierzchołek. NN(v_2) jest bliższym sąsiadem v_1 niż NN(v_1).

Co więcej, jeśli $d(v_1, NN(v_2)) \le d(v_1, NN(v_1))$, to

$$d(q, NN(v_1)) = d(v_1, NN(v_1)) + d(v_1, q) \ge d(v_1, NN(v_2)) + d(v_1, q) \ge d(q, NN(v_2)). \quad (5.2)$$

Wnioskiem z przykładu i wzoru (5.2) jest więc stwierdzenie, iż obie wersje algorytmu zwracają na wyjściu najbliższego sąsiada danej próbki. Algorytmy nie są jednak równoważne, jeśli chodzi o koszt przetwarzania wstępnego. Wbrew może pierwszemu wyobrażeniu, znalezienie NS tylko dla konkretnej „ćwiartki" przestrzeni nie jest (w najgorszym przypadku) szybsze niż znalezienie NS w całym zbiorze *P* (nie wydaje się możliwe takie zindeksowanie zbioru *P*, aby znalezienie NS było deterministycznie subliniowe w *n* dla dowolnego *d*). Wadą wersji alternatywnej (tj. z najbliższymi sąsiadami szukanymi dla „ćwiartek") jest natomiast związanie z każdym wierzchołkiem aż 2^d sąsiadów — w oryginalnej wersji z jednym wierzchołkiem skojarzony jest jeden sąsiad. Konkluzja jest oczywista: lepsza jest wersja pierwsza (wersja alternatywna została przedstawiona tylko dlatego, że idea metody jest w niej lepiej widoczna).

Z definicji algorytmu (wersja oryginalna) wynika, że czas szukania wynosi $O(d \log n + d \cdot 2^d)$, zaś przetwarzanie wstępne trwa $O(n^{d+1} \cdot d)$ i wymaga $O(n^d)$ pamięci.

Kompromis

Istnieje prosty kompromis pomiędzy kosztami wstępnej obróbki a szybkością szukania NS. Sortujemy zbiór *P* *d*-krotnie, kolejno według każdej z *d* współrzędnych, a następnie zamiast „pełnej" dekompozycji przestrzeni, przeprowadzamy hiperpowierzchnie jedynie przez co *k*-ty punkt ze zbioru *P*, w sensie opisanego porządku. Różnica polega na tym, że obecnie musimy dodatkowo policzyć odległości do *k*–1 dodatkowych punktów dla każdego z *d* wymiarów (zacienione pasy na Rys. 5.1 nie są już w kompromisowym rozwiązaniu puste). Koszt szukania wynosi zatem $O(d \log(\frac{n}{k}) + d \cdot 2^d + d^2 \cdot k)$, jednak koszt pamięciowy wstępnego przetwarzania maleje do $O((\frac{n}{k})^d)$. Odpowiednio maleje też czas wstępnego przetwarzania.

5.4. Wyniki eksperymentów i dyskusja

Opisany algorytm został zaimplementowany w języku C++ przy użyciu kompilatora g++ 2.95.3. Komputer, na którym wykonane zostały testy, to PC z dwoma procesorami Celeron 533 (wykorzystywany był jednak tylko jeden procesor), wyposażony w 384 MB pamięci i uruchomiony pod systemem Linux 2.4.

Tab. 5.2–5.14 przedstawiają wyniki testów na zbiorach danych o różnej liczności i dla różnych wartości współczynnika kompromisu *k*. Wartości *k* zostały dobrane tak, aby wystarczyło pamięci, a czas przetwarzania wstępnego był „rozsądnie" krótki. Zbiory danych w Tab. 5.2–5.13 zostały wygenerowane losowo z rozkładem jednostajnym na hiperkwadracie. Liczności zbiorów odniesienia są podane przy każdej tabeli, natomiast zbiory testowe liczyły zawsze po 100000 próbek (wygenerowanych w analogiczny sposób). W Tab. 5.14 podano wyniki na zbiorze odniesienia Iris, jednak i tu zbiór testowy stanowiło 100000 losowych próbek.

Kolejne wiersze w Tab. 5.2–5.14 podają odpowiednio: ilość pamięci zajmowanej przez struktury danych, czas alokacji tych struktur, czas zasadniczego wstępnego przetwarzania, czas szukania NS dla wszystkich próbek zbioru testowego przy użyciu zaprezentowanego algorytmu oraz czas szukania siłowego („naiwnego") (*brute force*).

Tabela 5.2: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 2-wymiarowy liczący 500 próbek.

	k=4	*k*=5	*k*=10	*k*=20	*k*=50
pamięć [MB]	0,24	0,16	0,04	0,01	0,01
alokacja [s]	0,03	0,01	< 0,01	< 0,01	< 0,01
preprocess [s]	0,74	0,48	0,11	0,03	0,01
szukanie [s]	0,82	0,75	0,65	0,65	1,09
brute-force [s]	3,17				

Tabela 5.3: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 2-wymiarowy liczący 1000 próbek.

	k=10	*k*=15	*k*=20	*k*=30	*k*=40
pamięć [MB]	0,16	0,08	0,05	0,03	0,02
alokacja [s]	0,03	0,01	0,01	< 0,01	< 0,01
preprocess [s]	0,87	0,39	0,22	0,10	0,06
szukanie [s]	0,83	0,84	0,87	0,91	1,07
brute-force [s]	6,93				

Tabela 5.4: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 2-wymiarowy liczący 10000 próbek.

	k=30	*k*=40	*k*=50	*k*=100
pamięć [MB]	1,78	1,03	0,69	0,23
alokacja [s]	0,17	0,13	0,05	0,03
preprocess [s]	237,82	140,17	87,14	20,64
szukanie [s]	4,29	4,46	4,97	8,21
brute-force [s]	197,40			

Tabela 5.5: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 3-wymiarowy liczący 100 próbek.

	k=4	*k*=5	*k*=10	*k*=20	*k*=50
pamięć [MB]	0,48	0,25	0,03	< 0,01	< 0,01
alokacja [s]	0,05	0,02	< 0,01	< 0,01	< 0,01
preprocess [s]	0,24	0,14	0,02	< 0,01	< 0,01
szukanie [s]	0,60	0,59	0,70	0,94	1,64
brute-force [s]	0,89				

Tabela 5.6: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 3-wymiarowy liczący 500 próbek.

	k=3	*k*=5	*k*=10	*k*=20	*k*=50
pamięć [MB]	142,14	30,52	3,82	0,48	0,04
alokacja [s]	8,48	2,02	0,29	0,05	< 0,01
preprocess [s]	274,10	56,80	6,36	0,79	0,06
szukanie [s]	1,55	1,23	1,02	1,12	1,80
brute-force [s]	3,69				

Tabela 5.7: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 3-wymiarowy liczący 1000 próbek.

	k=6	*k*=10	*k*=20	*k*=30	*k*=40	*k*=50	*k*=100
pamięć [MB]	142,14	30,53	3,83	1,21	0,49	0,26	0,04
alokacja [s]	8,72	1,90	0,23	0,09	0,03	0,01	< 0,01
preprocess [s]	480,96	99,11	12,30	3,92	1,60	0,83	0,12
szukanie [s]	1,66	1,31	1,36	1,54	1,77	2,16	3,40
brute-force [s]	7,79						

Tabela 5.8: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 3-wymiarowy liczący 10000 próbek.

	k=130	*k*=150	*k*=250	*k*=500
pamięć [MB]	14,05	9,29	2,07	0,36
alokacja [s]	0,90	0,57	0,10	0,02
preprocess [s]	1001,62	691,67	150,54	19,50
szukanie [s]	16,24	18,16	29,51	58,18
brute-force [s]	207,99			

Tabela 5.9: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 4-wymiarowy liczący 500 próbek.

	k=15	*k*=20	*k*=50	*k*=80
pamięć [MB]	81,57	23,85	0,62	0,15
alokacja [s]	2,75	0,82	< 0,01	< 0,01
preprocess [s]	92,09	27,88	0,85	0,25
szukanie [s]	1,68	1,71	2,97	4,19
brute-force [s]	4,65			

Tabela 5.10: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 4-wymiarowy liczący 1000 próbek.

	k=24	*k*=30	*k*=50	*k*=100
pamięć [MB]	189,94	81,58	9,78	0,63
alokacja [s]	6,76	2,88	0,30	0,02
preprocess [s]	387,30	168,91	21,90	1,60
szukanie [s]	2,20	2,45	3,62	5,66
brute-force [s]	9,63			

Tabela 5.11: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 4-wymiarowy liczący 10000 próbek.

	k=450	*k*=500	*k*=800
pamięć [MB]	17,24	9,92	1,90
alokacja [s]	0,57	0,33	0,06
preprocess [s]	911,20	525,26	126,67
szukanie [s]	83,37	93,43	171,66
brute-force [s]	262,96		

Tabela 5.12: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 5-wymiarowy liczący 500 próbek.

	k=30	*k*=50	*k*=80
pamięć [MB]	173,33	12,22	2,06
alokacja [s]	3,74	0,27	0,04
preprocess [s]	138,35	11,36	2,25
szukanie [s]	3,25	4,63	6,34
brute-force [s]	5,58		

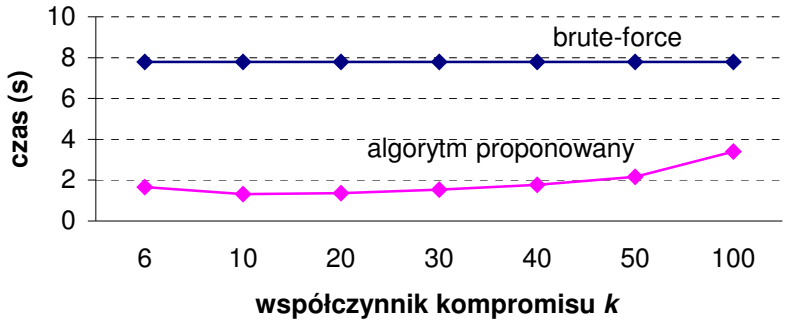
Tabela 5.13: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym. Zbiór odniesienia 5-wymiarowy liczący 1000 próbek.

	k=60	*k*=70	*k*=100	*k*=200	*k*=400
pamięć [MB]	173,34	92,72	12,23	0,40	0,05
alokacja [s]	4,18	2,11	0,28	0,01	< 0,01
preprocess [s]	257,81	140,76	21,40	1,01	0,14
szukanie [s]	5,91	6,41	8,72	16,11	27,82
brute-force [s]	11,38				

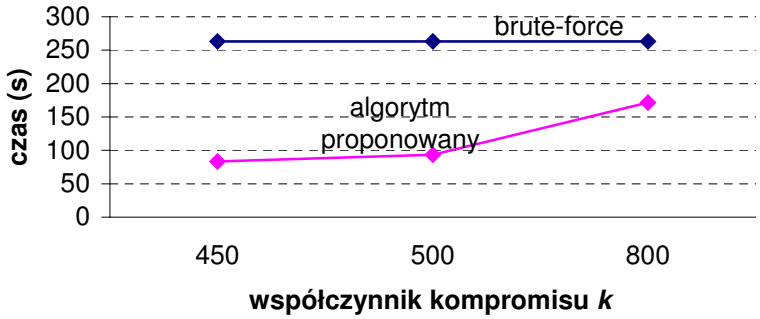
Tabela 5.14: Porównanie proponowanego algorytmu szukania NS z szukaniem siłowym.
Zbiór odniesienia Iris: 4 wymiary, 150 próbek.

	k=4	*k*=5	*k*=10	*k*=15	*k*=20	*k*=25	*k*=30	*k*=40	*k*=50
pamięć [MB]	127,27	49,44	3,09	0,61	0,25	0,08	0,04	0,02	0,01
alokacja [s]	4,58	1,68	0,13	0,03	0,01	0,01	< 0,01	< 0,01	< 0,01
preprocess [s]	55,92	21,64	1,42	0,30	0,14	0,05	0,02	0,01	< 0,01
szukanie [s]	0,83	0,83	0,98	1,16	1,27	1,53	1,77	2,07	2,55
brute-force [s]	1,43								

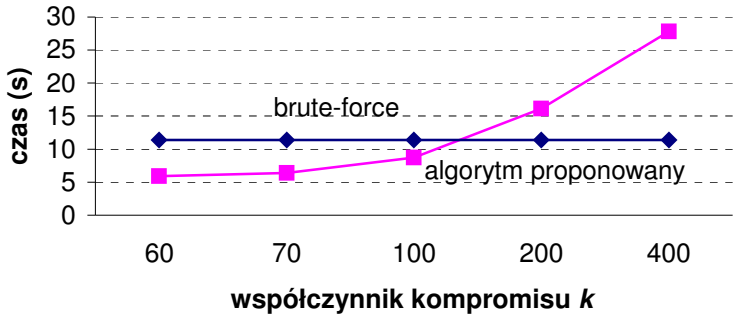
Rys. 5.3–5.5 prezentują graficzne porównanie czasów szukania proponowanego algorytmu i podejścia siłowego dla wybranych zbiorów odniesienia.



Rys. 5.3. Zależność czasu szukania od współczynnika kompromisu *k*.
Zbiór odniesienia 3-wymiarowy liczący 1000 próbek.



Rys. 5.4. Zależność czasu szukania od współczynnika kompromisu *k*.
Zbiór odniesienia 4-wymiarowy liczący 10000 próbek.



Rys. 5.5. Zależność czasu szukania od współczynnika kompromisu *k*.
Zbiór odniesienia 5-wymiarowy liczący 1000 próbek.

Jak widać, zwiększenie *k* powoduje zmniejszenie obciążenia pamięciowego oraz skrócenie przetwarzania wstępnego za cenę wolniejszego szukania. Tym niemniej, w wielu przypadkach (p. np. Tab. 5.2 lub 5.5) zaobserwowano dziwny efekt skracania do pewnego momentu czasu szukania wraz ze wzrostem *k* (przy odpowiednio dużych wartościach *k* efekt ten znika i czas szukania zgodnie z teorią rośnie). Przypuszczalnie jest to spowodowane *cache*'m procesora: przy małych wartościach *k* posortowane tablice wartości poszczególnych cech mogą nie mieścić się w całości w *cache*'u, a zatem dostęp do nich jest stosunkowo wolny. Warto zauważyć, że zjawisko to nie zostało odnotowane dla przypadków w wyższych wymiarach (4–5), co zapewne wiąże się z faktem, iż dla tych zbiorów testowanie algorytmu z niskimi wartościami *k* było niemożliwe z uwagi na ogromne koszta obróbki wstępnej.

Warto zwrócić uwagę, że przedstawiony algorytm przy odpowiednio niskim wymiarze (2–3) nadaje się także do zastosowań *on-line*, gdy celem jest minimalizacja sumy czasu wstępnego przetwarzania i zasadniczego szukania. Zauważmy, że jeśli (z grubsza mówiąc) $k > n^{(d-1)/d}$, to czas generacji struktury danych jest krótszy od czasu znalezienia NS dla każdej próbki zbioru odniesienia metodą brute-force. Obrazuje to np. przypadek $k = 40$ w Tab. 5.3; czas wstępnego przetwarzania wynosi 0,060 s, natomiast szukanie NS metodą brute-force dla 1000 próbek trwa nieco dłużej, bo ok. 0,069 s. W wielu przypadkach niskowymiarowych istnieją jeszcze większe wartości *k* takie, iż suma czasu wstępnej obróbki i czasu szukania NS dla wszystkich próbek zbioru odniesienia będzie niższa od czasu szukania najbliższego sąsiada dla wszystkich próbek metodą brute-force.

Jakie może być konkretne zastosowanie tej właściwości algorytmu? W pracy (Strzecha, 2001) przedstawiono metodę segmentacji obrazu dla potrzeb wysokotempe-

raturowych pomiarów pewnych własności fizykochemicznych wybranych materiałów; algorytm Strzechy wykorzystuje regułę *k*-NN dla próbek w przestrzeni zaledwie dwóch cech. W przypadku, gdy liczba sąsiadów wynosi 1, można łatwo wykorzystać w tym zastosowaniu zaproponowaną metodę.

Przedstawiony algorytm można użyć także do szybkiego szukania *k* scentrowanych sąsiadów (p. Rozdz. 7) w metryce miejskiej. W klasyfikatorze *k*-NCN dla każdego z kolejno szukanych sąsiadów możliwe jest ustalenie jego optymalnego położenia, tj. takiego, dla którego środek ciężkości układu sąsiadów pokrywałby się z zadaną próbką testową. Dzięki temu, *i*-ty sąsiad scentrowany może być pojmowany jako najbliższy sąsiad punktu będącego „przeciwwagą" środka ciężkości sąsiadów o indeksach 1, ..., *i*–1, co umożliwia bezpośrednie zastosowanie zaproponowanego algorytmu.

Rozdział VI
Algorytmy redukcji dla reguły decyzyjnej 1-NN

Główną wadą reguły decyzyjnej typu *k* najbliższych sąsiadów (*k*-NN), a tym bardziej takich klasyfikatorów, jak opisywane w Rozdz. 7 *k*-NCN i *k*-NSN, jest stosunkowo wolna klasyfikacja, nie do przyjęcia w niektórych zastosowaniach, np. przy analizie obrazu metodą piksel po pikslu. Wadę tę w niemałym stopniu dziedziczy również najszybsza wersja reguły *k*-NN, tj. 1-NN. Czas klasyfikacji przy użyciu wymienionych reguł decyzyjnych zależy od wielkości zbioru odniesienia. Jest zatem jasne, iż redukcja zbioru odniesienia, tj. zastąpienie oryginalnego zbioru zbiorem mniejszym, lecz możliwie dobrze aproksymującym powierzchnie decyzyjne między klasami, przyniesie przyspieszenie klasyfikacji.

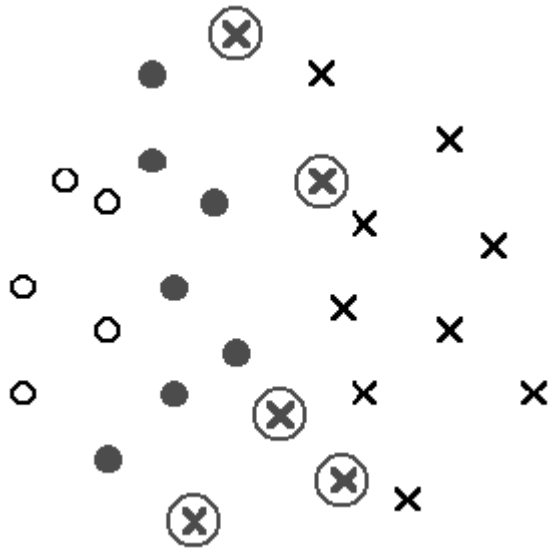
W rozdziale omówiono kryteria oceny przydatności poszczególnych algorytmów redukcji, opisano bliżej kilka najczęściej używanych i ważnych historycznie koncepcji, przedstawiono szereg algorytmów nowszych, a następnie zaproponowano opracowany przez autora rozprawy schemat z lokalnym wyborem zbioru zredukowanego, polegający na zastąpieniu tradycyjnie pojedynczego zbioru zredukowanego zespołem kilku (kilkunastu) takich zbiorów, z których dla danej próbki wybierany jest tylko jeden przy pomocy bardzo szybkiego kryterium (Grabowski, 2002c; Grabowski i Jóźwik, 2003). Wyniki eksperymentów, opisanych w Rozdz. 6.5, sugerują atrakcyjność przedstawionego podejścia.

Ponadto zaproponowano modyfikacje niektórych znanych algorytmów oraz przedstawiono i porównano empirycznie kilka heurystycznych implementacji tworzenia zbioru zredukowanego Tomeka.

6.1. Specyfika problemu i kryteria oceny algorytmów

Redukcja zbioru odniesienia ma sens zasadniczo dla reguły 1-NN. Redukcja powoduje „rozrzedzenie" zbioru, a zatem przy zastosowaniu np. reguły *k*-NN istniałoby poważne niebezpieczeństwo, iż część sąsiadów znajdzie się zbyt daleko od badanej próbki, aby dobrze przewidywać jej klasę. Tym niemniej, niekiedy — zwłaszcza na zbiorach zaszu-

mionych — algorytmy redukcji pierwotnie przewidziane dla reguły 1-NN adaptuje się do reguł *k*-NN z małymi wartościami *k*, np. 3 (Wilson i Martinez, 2000).



Rys. 6.1. Przykładowa redukcja zbioru 2-wymiarowego złożonego z 2 klas. Elementy wyróżnione („krzyżyki" z otoczką i wypełnione „kółka") wchodzą do zbioru zredukowanego.

W literaturze podaje się ponad 20 algorytmów redukcji zbioru odniesienia. W zdecydowanej większości algorytmów zredukowany zbiór jest pewnym podzbiorem zbioru oryginalnego (poglądową ilustrację efektu działania redukcji ukazuje Rys. 6.1). Większość algorytmów redukcji podawanych w literaturze produkuje zbiory zgodne z oryginalnymi zbiorami odniesienia. Zgodność oznacza, iż reguła 1-NN wykorzystująca zbiór zredukowany jako zbiór odniesienia poprawnie klasyfikuje wszystkie próbki ze zbioru oryginalnego. Kryterium zgodności ma być niejakim potwierdzeniem, iż algorytm redukcji produkuje zbiór zredukowany wyznaczający powierzchnie decyzyjne bliskie powierzchniom decyzyjnym zdefiniowanym w oparciu o oryginalny zbiór odniesienia. Jak pokazano w dalszej części rozdziału, przekonanie to jest złudne, gdyż algorytmy bazujące na kryterium zgodności są mocno podatne na przeuczenie. Szybkość redukcji ma zwykle znaczenie drugorzędne, gdyż proces ten wykonywany jest tylko raz dla danego zbioru. Tym niemniej, niektóre algorytmy wydają się zbyt wolne dla bardzo dużych zbiorów. Dla ogromnych zbiorów odniesienia, liczących setki tysięcy lub miliony obiektów, nawet najszybszy z algorytmów produkujących zbiór zgodny z oryginalnym, tj. algorytm „skondensowany najbliższy sąsiad" (*Condensed Nearest Neighbor*, CNN) (Hart, 1968), może być zbyt wolny. W takim przypadku rezygnacja z wymogu zgodności jest już koniecznością (Jóźwik i in., 1995).

Wiele algorytmów redukcji, zwłaszcza w klasie algorytmów spełniających wymóg zgodności, działa w sposób przyrostowy albo — przeciwnie — poprzez kolejną eliminację zbędnych elementów. W pierwszym przypadku procedura redukcji startuje z pustego zbioru zredukowanego i sukcesywnie go powiększa (do liczności zadanej przez użytkownika albo wyznaczonej automatycznie). Algorytm redukcji przez eliminację zaczyna od kompletnego zbioru odniesienia i po kolei usuwa elementy, aż do spełnienia kryterium stopu.

Poniżej omówiono systematycznie kryteria decydujące o przydatności danego algorytmu redukcji.

Szybkość klasyfikacji. Jest to jedno z najważniejszych kryteriów. Szybkość klasyfikacji jest ściśle związana z wielkością zbioru zredukowanego. Nie musi to być jednak zależność wprost proporcjonalna; może być subliniowa, jeśli oprócz redukcji używa się odpowiedniego algorytmu szybkiego szukania NS (p. Rozdz. 5).

Pamięć dla klasyfikatora (tj. dla zbioru zredukowanego). Wielkość pamięci przechowującej zbiór zredukowany wiąże się z szybkością klasyfikacji. Samo obciążenie pamięciowe nie jest obecnie już tak dotkliwe jak przed laty, ale również może mieć znaczenie przy bardzo dużych zbiorach. Warto zaznaczyć, iż w niektórych algorytmach, np. z pracy (Wettschereck i Dietterich, 1995), oprócz próbek, zbiór zredukowany może zawierać bardziej złożone obiekty (np. hiperprostokąty); ich reprezentacja będzie zapewne bardziej kosztowna pamięciowo niż w przypadku punktów.

Jakość klasyfikacji (ogólna). Kolejne bardzo ważne kryterium. Przy redukcji estymuje się ją przeważnie z użyciem zbioru testowego. Na ogół po redukcji jakość klasyfikacji nieco się pogarsza. Może jednak się poprawić: próbki z oryginalnego zbioru ewidentnie „źle leżące" (tj. szum) mogą zostać usunięte i tym samym granice między klasami wygładzone.

Jakość klasyfikacji w obecności szumu. Obecność szumu stwarza dwa podstawowe niebezpieczeństwa. Pierwsze polega na tym, że niewiele próbek zostanie usuniętych z oryginalnego zbioru, gdyż algorytm będzie starał się możliwie wiernie oddać zaszumione (czyli w sztuczny sposób skomplikowane) granice decyzyjne między klasami (zjawisko przeuczenia). Drugi problem to możliwość usunięcia „dobrych" próbek, a pozostawienia „złych", tj. będących szumem. Przykładowo, jest na to narażony

algorytm CNN (Hart, 1968). Algorytmy słabo radzące sobie z opisanymi problemami (mówi się: wrażliwe na szum) na ogół oferują niską jakość klasyfikacji.

Pomocnymi „narzędziami" do usuwania szumu mogą być wstępna selekcja cech oraz reklasyfikacja lub odfiltrowanie zbioru odniesienia (techniki te zostały omówione w dalszej części rozdziału).

Szybkość uczenia (generacji zbioru zredukowanego). Kryterium mniej ważne, gdyż proces uczenia wykonywany jest tylko raz. W praktyce jednak, zbyt wolne algorytmy nie mogą być zastosowane na bardzo dużych zbiorach. Niestety, redukcja jest najbardziej potrzebna właśnie w przypadku bardzo dużych zbiorów, zatem warto dbać o optymalizację i tego kryterium.

Pamięć potrzebna do uczenia. Na ogół koszt pamięciowy uczenia jest niski i wynosi $O(n)$, n – liczność oryginalnego zbioru, a nawet — jeśli dane wejściowe podawane są kolejno (np. w czasie trwania eksperymentu) i nie muszą być wszystkie przechowywane w pamięci — może wynosić tylko $O(h)$, h – wielkość zbioru zredukowanego (tak będzie w przypadku algorytmu IB2 (Kibler i Aha, 1987), będącego *de facto* pierwszym cyklem algorytmu Harta).

Z drugiej strony, w przypadku algorytmu selektywnego (Ritter i in., 1975) do uczenia potrzebna jest macierz binarna $n \times n$, co może wykluczyć tę metodę z niektórych zastosowań.

Zdolność szybkiej modyfikacji zbioru zredukowanego w przypadku dynamicznym. Jeśli już po wygenerowaniu zbioru zredukowanego oryginalny zbiór uczący zostanie zmieniony (przybędą nowe próbki i/lub zostaną usunięte niektóre inne próbki), to zignorowanie tego faktu w oczywisty sposób prowadzić musi do utraty szansy poprawy jakości klasyfikacji. Dlatego też w takiej sytuacji zbiór zredukowany powinien zostać uaktualniony. Najniżej zostanie oceniony algorytm, dla którego usunięcie bądź dodanie jednej próbki do zbioru oryginalnego wymaga generacji zbioru zredukowanego „od zera".

6.2. Przegląd metod

W podrozdziale opisano bliżej kilka najbardziej znanych algorytmów tworzących zbiór zredukowany zgodny z oryginalnym zbiorem odniesienia, a następnie wspomniano o innych metodach redukcji.

Algorytm Harta (CNN)

Koncepcja przyrostowego budowania zbioru zredukowanego pojawiła się w pierwszym historycznie algorytmie *Condensed Nearest Neighbor* (CNN) (Hart, 1968), co można przetłumaczyć jako „skondensowany najbliższy sąsiad". Na początku zbiór zredukowany jest pusty. Pierwsza próbka w zbiorze jest dołączana do zbioru zredukowanego. Następnie przegląda się po kolei wszystkie pozostałe próbki, i jeżeli niektóre z nich są źle rozpoznawane, tj. źle klasyfikowane regułą 1-NN, przez aktualny zbiór zredukowany, to zostają (automatycznie po stwierdzeniu błędnego rozpoznania) dołączone do zbioru zredukowanego. Prezentację całego oryginalnego zbioru odniesienia nazwiemy cyklem. Po zakończeniu pierwszego cyklu opisana procedura zostaje powtórzona, z tą różnicą, iż klasyfikowane są teraz tylko te punkty, które nie zostały wcześniej dołączone do zbioru zredukowanego. Proces jest kontynuowany tak długo, aż w którymś cyklu nie zostanie dołączony już żaden nowy element do zbioru zredukowanego.

Jako ciekawostkę można dodać, iż idea Harta była dwukrotnie „odkrywana" ponownie: jako wspomniany już algorytm IB2 (Kibler i Aha, 1987) oraz jako *Grow and Learn* (Alpaydin, 1990).

W artykule (Grabowski, 1999) autor niniejszej rozprawy przedstawił prostą metodę implementacji algorytmu Harta o złożoności $O(dnh)$, d – wymiar przestrzeni (tj. liczba cech), n – liczność oryginalnego zbioru, h – liczność zbioru zredukowanego. Bazuje ona na obserwacji, że choć w praktyce cykli algorytmu Harta jest zaledwie kilka, to w najgorszym przypadku, w implementacji „naiwnej", może ich być n.

W prostej implementacji w każdym cyklu liczone są odległości od każdego testowanego obiektu do wszystkich obiektów z aktualnego zbioru zredukowanego. Koszt algorytmu w implementacji naiwnej w najgorszym przypadku wynosi zatem $O(dn^3)$. Oto przykład oryginalnego zbioru S (2 klasy, 2 cechy: x i y, $n = 2k$ obiektów), dla którego sytuacja taka ma miejsce (Rys. 6.2).

indeks	klasa	x	y
1	1	0	0
2	2	0	1
3	1	1	1/2 – 1/8
4	2	2	1/2 + 1/8
5	1	4	1/2 – 1/16
6	2	8	1/2 + 1/16
.....			
$2k$–1	1	2^{2k-1-3}	$1/2 - 1/2^{k+1}$
$2k$	2	2^{2k-3}	$1/2 - 1/2^{k+1}$

Rys. 6.2. Przykładowy 2-wymiarowy zbiór odniesienia, dla którego koszt naiwnej implementacji algorytmu redukcji Harta wynosi $O(dn^3)$, gdzie $n = 2k$

Proszę zwrócić uwagę na ujemny znak we współrzędnej *y* próbki o indeksie 2*k*.

Załóżmy, że użyta jest metryka euklidesowa. W pierwszym cyklu do zbioru zredukowanego zaliczone będą obiekty o indeksach 1, 2 i 2*k*. W drugim cyklu obiekty 3, 4, ..., 2*k*–2 będą poprawnie zaklasyfikowane, a jedynym dołączonym obiektem będzie obiekt 2*k*–1. W trzecim cyklu dołączony będzie obiekt o indeksie 2*k*–2 itd. aż do cyklu (2*k*–3)-ciego. Średnio, w naiwnej implementacji, w każdym cyklu liczy się $O(k^2)$ odległości, a zważywszy, że $k = n/2$ oraz że liczba cykli wynosi *n*–3, otrzymujemy $O(n^3)$ odległości do policzenia.

Zaprezentowana poniżej idea zmniejszająca liczbę obliczanych odległości jest bardzo prosta: po negatywnej klasyfikacji danego obiektu w danym cyklu należy zapamiętać (uaktualnić) dla niego aktualną liczność zbioru zredukowanego. W kolejnym cyklu dla danego obiektu wystarczy policzyć jedynie odległości do nowo dołączonych „w międzyczasie" obiektów zbioru odniesienia. Nietrudno zauważyć, że w całej procedurze redukcji należy obecnie policzyć ok. $n \cdot h$ odległości, niezależnie od liczby cykli. W praktyce osiągana oszczędność czasu jest przeważnie dwu-trzykrotna.

Algorytm Gowdy–Krishny

Wyniki algorytmu Harta zależą od kolejności prezentowanych obiektów. Cechę tę niekoniecznie należałoby postrzegać jako wadę — niestety, w pierwszych krokach CNN do zbioru zredukowanego dołączane są często obiekty dość przypadkowe, położone daleko od hiperpowierzchni rozdzielających klasy. Próbą usunięcia tej wady był algorytm Gowdy–Krishny (oznaczany dalej G–K) (Gowda i Krishna, 1979). Jest on

właściwie procedurą Harta poprzedzoną odpowiednim uporządkowaniem zbioru odniesienia. Konkretnie, elementy zbioru są posortowane wg rosnących wartości pewnej miary, zwanej pozycyjną, która w pewnym sensie określa położenie obiektu względem obiektów z innych klas. Mówiąc w uproszczeniu, uporządkowanie to przemieszcza na początek zbioru obiekty leżące blisko granic między klasami. Tym samym, próbki położone daleko od granic decyzyjnych mają stosunkowo małe szanse wejścia do zbioru zredukowanego.

Jest oczywiste, że algorytm G–K, podobnie jak CNN, produkuje zbiór zgodny ze zbiorem oryginalnym.

Algorytm Tomeka

Metoda opisana w (Tomek, 1976c) włącza do zbioru zredukowanego wszystkie pary (zwane dalej Tomekowymi) obiektów x i y z różnych klas, takich że wnętrze kuli rozpiętej na x i y nie zawiera żadnego innego obiektu z oryginalnego zbioru. Można w tym miejscu zwrócić uwagę, że pary Tomekowe odpowiadają tym krawędziom w grafie Gabriela (który zostanie zaprezentowany w Rozdz. 7), które łączą obiekty z różnych klas. Żadna próbka nie spełniająca przestawionego warunku nie wchodzi do zbioru odniesienia. Algorytm Tomeka, wbrew intencji autora, nie zawsze jednak produkuje zbiór zgodny (Toussaint, 1994) — kontrprzykład na Rys. 6.3. Oryginalny algorytm Tomeka, przy założeniu metryki euklidesowej, utworzy zbiór zredukowany złożony tylko z obiektów 1 i 2. Obiekt 4 będzie źle klasyfikowany.

Usterkę tę, rzadko ujawniającą się w praktyce, łatwo usunąć, np. dodając do zbioru zredukowanego nie spełniającego warunku zgodności po kolei obiekty źle klasyfikowane, zaczynając od obiektu o najmniejszym stopniu mylnej klasyfikacji, którego miarą jest różnica odległości do najbliższego sąsiada z tej samej klasy i odległości do najbliższego sąsiada z innej klasy (Grabowski i Jóźwik, 1999).

Rys. 6.3. Przykład zbioru odniesienia, dla którego algorytm redukcji Tomeka nie tworzy zbioru zredukowanego zgodnego z oryginalnym

Cechą algorytmu Tomeka jest lokalnie wierne zachowywanie oryginalnych granic decyzyjnych, dlatego błąd reguły 1-NN dla zbioru zredukowanego tym algorytmem zwykle jest bardzo zbliżony do błędu na oryginalnym zbiorze. Pod tym względem algorytm Tomeka należy do ścisłej czołówki algorytmów redukcji, które zwykle powodują jednak niewielkie pogorszenie jakości. Niestety, stopień redukcji algorytmu Tomeka jest jednocześnie niewielki, zwykle mniej niż dwukrotny. Tym niemniej, jeśli jakość klasyfikacji w danym zastosowaniu ma wyraźnie wyższy priorytet niż szybkość, to algorytm Tomeka może być rozważany.

W pracy (Grabowski, 2000a) przedstawiono szereg wersji implementacyjnych generacji zbioru Tomeka. Należy podkreślić, że we wszystkich poniższych rozważaniach związanych z algorytmem Tomeka przyjęta została metryka euklidesowa, zalecana dla tego algorytmu w oryginalnej pracy (Tomek, 1976c). Niżej opisano owe implementacje i przeprowadzono eksperyment na rzeczywistych zbiorach dwudecyzyjnych. Implementację A można uznać za „naiwną".

A. Przeglądane są wszystkie pary obiektów z różnych klas, z unikaniem powtórzeń (np. jeśli była sprawdzona para (1,5), to omija się już (5,1)) oraz z przejściem do następnej pary, jeśli oba bieżące obiekty już są składowymi jakichś znalezionych par Tomekowych. Przy sprawdzaniu, czy bieżąca próbka należy do wnętrza danej kuli, pomija się liczenie pierwiastka.

B. Została dodana heurystyka szukania NS. Ideę tę przedstawiono w pracy (Grabowski, 1999) i w Rozdz. 4.2 niniejszej rozprawy. W tym przypadku nie można już zrezygnować z obliczania pierwiastka we wzorze na odległość.

C. Na początku wykonywana jest procedura Harta (z metryką euklidesową i odległością dla Harta liczoną bez pierwiastka); obiekty ze zbioru zredukowanego Harta przenoszone są na początek oryginalnego zbioru. Na tak uporządkowanym zbiorze uruchamiana jest procedura B. Ponieważ punkty otrzymane algorytmem Harta leżą w większości blisko granic między klasami, zatem uzasadniona jest nadzieja, iż przy niniejszym porządku szybciej zostanie znaleziona próbka falsyfikująca testowaną parę.

D. Podobnie jak w implementacji C, jedynie z różnicą w procedurze Harta. Po pierwszym cyklu algorytmu Harta, obiekty aktualnego zbioru zredukowanego są

ustawiane w odwrotnej kolejności, tzn. obiekt dołączony jako ostatni będzie w drugim cyklu prezentowany jako pierwszy. Dalej algorytm Harta jest już wykonywany bez zmian.

E. Podobnie jak w implementacji D, ale z dodaną heurystyką (p. punkt B) szukania NS dla Harta. Tym razem nie można już pominąć liczenia pierwiastka w procedurze Harta.

F. Podobnie jak w implementacji E, ale z trzema (zamiast jednej) pomocniczymi tablicami odległości do wybranych próbek przy heurystyce szukania NS. Przyjęte ustalenia: jeżeli cech jest więcej niż 10, to wykonywane są maksymalnie trzy próby „odrzutu" bieżącej próbki, przy liczbie cech od 6 do 10 włącznie — maksymalnie dwie próby, a przy co najwyżej 5 cechach — jedna próba.

G. Na bazie implementacji B, ale z istotną modyfikacją. Dla danej próbki X brana jest pierwsza napotkana próbka Y z innej klasy, a następnie sprawdza się, które obiekty Z, z innej klasy niż próbka X, tworzą rozwarty kąt XYZ — należy zauważyć, że przy rozwartym kącie XYZ niemożliwa jest para Tomekowa (X, Z).

H. Podobnie jak w implementacji G, z tą różnicą, iż Y nie jest dla X pierwszą napotkaną próbką z innej klasy, a najbliższą z k pierwszych napotkanych próbek z innej klasy. Przyjęto $k = 5$.

W Tab. 6.1 przedstawiono czasy generacji zbiorów Tomeka dla implementacji opisanych w punktach od A do H dla trzech par klas zbioru FerritesOld dotyczącego kontroli jakości rdzeni ferrytowych. Każda para klas zawiera 400 punktów (po 200 na klasę); są to zbiory otrzymane po selekcji cech metodą FSS. W całym zadaniu było pięć klas, a zatem 10 par klas. Wybrano trzy pary najbardziej reprezentatywne: dla pary 1 / 2 błąd jest zerowy, a liczność zbioru zredukowanego ekstremalnie niska (dwa elementy), para 3 / 4 była najbardziej zaszumiona i w konsekwencji wygenerowany dla niej zbiór Tomeka był największy ze wszystkich 10 zbiorów, wreszcie liczność zbioru Tomeka dla pary 2 / 5 była najbardziej zbliżona do mediany liczności dla wszystkich par klas. Tym samym, zamieszczone wyniki oddają szybkość różnych implementacji przy różnym charakterze poddawanego redukcji zbioru. Dla uzyskania dokładniejszych

pomiarów, każda procedura (z pominięciem wejścia/wyjścia) została wykonana 30 razy w pętli.

Najszybsza okazała się metoda D. Jak widać, metoda D jest ponad 2-krotnie szybsza od A (zaś w skrajnym przypadku pary 1 / 2 różnica prędkości jest ok. 6-krotna).

Tabela 6.1: Porównanie szybkości ośmiu implementacji oryginalnego algorytmu Tomeka

para klas	liczba cech	licz. zb. zred.	czas redukcji (s)							
			A	B	C	D	E	F	G	H
1 / 2	1	2	0,35	0,33	0,08	0,06	0,06	0,06	0,26	0,29
2 / 5	8	106	0,31	0,17	0,15	0,14	0,15	0,16	0,21	0,20
3 / 4	5	331	0,63	0,53	0,37	0,25	0,28	0,29	0,79	0,72

Algorytm selektywny (SNN)

Algorytm selektywny (*Selective Nearest Neighbor*, SNN), zaproponowany w (Ritter i in., 1975), narzuca na zbiór zredukowany warunek silniejszy od zgodności, a mianowicie dla każdego obiektu ze zbioru oryginalnego, jego najbliższy sąsiad ze zbioru zredukowanego musi leżeć w mniejszej odległości niż jakikolwiek obiekt z innej klasy ze zbioru oryginalnego. Gdyby w powyższym sformułowaniu zmienić ostatnie słowo na „zredukowanego", otrzymałoby się definicję zgodności. Wymóg nałożony przez algorytm selektywny pozwala jednak na przeprowadzenie redukcji przy użyciu eleganckiej procedury gwarantującej osiągnięcie minimalnego zbioru o zadanej właściwości. Niestety, ów minimalny zbiór nie jest określony jednoznacznie.

Zasadnicza procedura SNN polega na usuwaniu z macierzy binarnej wierszy i kolumn reprezentujących próbki ze zbioru oryginalnego. Wiersze (jeszcze nieusunięte) to kandydaci do zbioru zredukowanego. Kolumny to próbki, które zbiór zredukowany musi dobrze rozpoznać.

Pełny opis algorytmu można znaleźć w oryginalnej pracy (Ritter i in., 1975). W przeprowadzonych przez autora rozprawy testach, wykorzystano uproszczoną wersję SNN. Uproszczenie polegało na pominięciu ostatniego, piątego kroku, który jest zdecydowanie najbardziej złożony, a jego wpływ na końcowy wynik — najmniejszy (a często żaden, co zaznaczają sami autorzy algorytmu).

Algorytm selektywny ma bardzo czasochłonną, bo aż $O\left(n^4 d\right)$ wynoszącą, złożoność w najgorszym przypadku. W praktyce jednak tworzenie zbioru zredukowanego jest zwykle szybsze.

Modyfikacja algorytmu selektywnego (SNN-CR) (Grabowski, 2000a)

Algorytm SNN korzysta z macierzy binarnej, w której jedynka na przecięciu wiersza *i* oraz kolumny *j* oznacza, że próbka *i* należy do tej samej klasy, co próbka *j* i jest ona bliższym sąsiadem próbki *j* niż jakakolwiek próbka z innej klasy. W procedurze SNN dany wiersz *i* jest usuwany, jeżeli istnieje inny wiersz *j* posiadający jedynki na wszystkich pozycjach jedynek w *i*. Należy zauważyć, że w szczególnym przypadku, gdy wiersze *i* oraz *j* są jednakowe, można usunąć dowolny z nich (algorytm tego nie precyzuje). W zmodyfikowanej wersji, którą autor rozprawy nazwał algorytmem selektywnym z ostrożnym usuwaniem próbek (*Selective Nearest Neighbor — Careful Removal*, SNN-CR), usuwa się ten z pary jednakowych wierszy, który reprezentuje próbkę położoną w dalszej odległości od swego najbliższego sąsiada z innej klasy. Innymi słowy, usuwane są raczej próbki „z głębi" zbioru niż położone blisko granic decyzyjnych. Odwrotną nierówność przyjęto natomiast przy usuwaniu kolumn w analogicznej sytuacji.

Algorytm Gatesa (RNN)

Wczesną realizacją koncepcji generowania zbioru zredukowanego poprzez usuwanie kolejno zbędnych elementów był algorytm RNN (*Reduced Nearest Neighbor*) (Gates, 1972). Pierwszym krokiem jest procedura redukcji Harta. Następnie punkty ze zbioru Harta są przeglądane po kolei, i jeżeli usunięcie danego punktu nie prowadzi do mylnej klasyfikacji pozostałych próbek ze zbioru odniesienia, to dany punkt należy usunąć. Rozpatrzenie ostatniego punktu kończy pierwszy cykl. Cykl opisany powtarzamy (dla punktów, które jeszcze zostały w zbiorze zredukowanym) tak długo, aż z otrzymanego zbioru nie można usunąć już żadnego obiektu. Warto zauważyć, że próbki usuwane nie muszą być klasyfikowane poprawnie, zatem algorytm oryginalny nie gwarantuje zgodności zbioru zredukowanego z pełnym zbiorem odniesienia. Niekiedy może to jednak być zaletą, gdyż wersja Gatesa może usuwać próbki zaszumione. Wadą algorytmu Gatesa jest stosunkowo długi czas przeprowadzania redukcji.

Algorytmy Skalaka

Skalak (1994) przedstawił dwa algorytmy redukcji — oryginalność obu polega na ich skrajnej prostocie. Pierwsza metoda (RMHC), używająca techniki określanej w literaturze poświęconej sztucznej inteligencji mianem *random mutation hill climbing* („wspinaczka po wzgórzu z losowymi mutacjami"), wybiera losowo *h* próbek ze zbioru

odniesienia i estymuje jakość takiego zbioru na pełnym zbiorze odniesienia z użyciem reguły 1-NN. Następnie w pętli wykonywanych jest m_1 tzw. *mutacji*. Mutacja polega na wylosowaniu jednej próbki z aktualnego zbioru zredukowanego oraz jednej próbki z tej części zbioru odniesienia, która nie należy do aktualnego zbioru zredukowanego. Jeśli zamiana tych próbek (tj. odrzucenie pierwszej i dodanie drugiej) prowadzi do polepszenia jakości klasyfikacji regułą 1-NN opartą na zbiorze zredukowanym, to mutacja zostaje przyjęta. Jeśli nie, wówczas mutację należy odrzucić, czyli powrócić do poprzedniego zbioru zredukowanego. Metoda Skalaka nie gwarantuje zgodności zbioru zredukowanego ze zbiorem oryginalnym, co więcej: w praktyce zwykle wygenerowany zbiór jest od zgodności odległy. Należy zauważyć, że algorytm zależny jest od dwóch wybranych przez użytkownika parametrów: h i m_1. Liczba mutacji m_1 mieści się zazwyczaj w przedziale 100–1000 (zbyt duża wartość może prowadzić do przeuczenia).

Autor rozprawy użył algorytmu redukcji RMHC do segmentacji barwnej jąder komórkowych w cytologicznych obrazach mikroskopowych (Bieniecki, Grabowski i in., 2002). Pomimo prawie 20-krotnego stopnia redukcji zbioru odniesienia, wizualny efekt segmentacji barwnej przy użyciu zbioru zredukowanego tylko nieznacznie różnił się od wyniku segmentacji przy użyciu pełnego zbioru odniesienia i reguły 1-NN, co w danym zastosowaniu było w pełni satysfakcjonujące.

W pracy (Grabowski, 2002c) zmodyfikowano algorytm RMHC. Modyfikacja polega na kontynuacji uczenia wg przedstawionej procedury. Następny krok polega mianowicie na próbie tworzenia nowych (sztucznych) prototypów w zbiorze zredukowanym metodą błądzenia przypadkowego (ang. *random walk*). Konkretnie, wykonuje się w pętli m_2 mutacji polegających na przesunięciu losowej próbki z aktualnego zbioru zredukowanego na jednej losowej cesze o +/– 0,5 (znak jest również losowany). Jeśli takie przesunięcie poprawia jakość estymowaną na zbiorze uczącym, to mutacja zostaje zaakceptowana. Warto zwrócić uwagę, że wartość 0,5 jest połową odchylenia standardowego dla każdej znormalizowanej cechy.

Druga z metod Skalaka (1994) jest jeszcze prostsza: m-krotnie losuje się h próbek ze zbioru uczącego i ostatecznie wybiera ten zbiór zredukowany, który osiągnął najwyższą jakość na zbiorze uczącym. I tu parametry (m i h) są zadawane przez użytkownika. W testach na kilku zbiorach UCI przeprowadzonych w oryginalnej pracy, oba algorytmy osiągały porównywalną jakość i jednocześnie średnio wyższą niż jakość 1-NN z pełnym zbiorem odniesienia, mimo iż zadany stopień redukcji był skrajnie wysoki.

Oprócz opisanych algorytmów, w literaturze pojawiło się wiele innych metod redukcji. Jedną z najwcześniejszych propozycji jest iteracyjny algorytm Swongera ICA (*Iterative Condensation Algorithm*) (1972). W metodzie tej stosuje się naprzemiennie cykle usuwania oraz dołączania próbek. Wynikowy zbiór zredukowany jest zgodny ze zbiorem oryginalnym.

Algorytm VSM (*Variable Similarity Metric*), zaproponowany w pracy (Lowe, 1995), jest złożonym systemem uczenia, dokonującym m. in. redukcji zbioru odniesienia (w uproszczeniu: usuwane są próbki posiadające najbliższych sąsiadów tylko z jednej klasy) oraz ważenia cech.

Podobną do użytej w schemacie VSM ideę redukcji zastosowano w pracy (Wu i in., 2002), osiągając kilkakrotne przyspieszenie klasyfikacji regułą *k*-NN przy zachowaniu wyjściowej jakości na bardzo dużych i wysokowymiarowych zbiorach dotyczących rozpoznawania pisma ręcznego.

Algorytm *Shrink* (Kibler i Aha, 1987) jest pewną modyfikacją algorytmu Gatesa; niestety, mniej odporną na szum niż jego pierwowzór.

Redukcja w złożonym systemie MCS (*Model Class Selection*), przedstawionym przez Brodley (1993), usuwa obiekt q z oryginalnego zbioru, o ile w większości przypadków jego klasa nie zgadza się z klasą obiektów, dla których q jest jednym z k najbliższych sąsiadów. Idea ta ma na celu eliminację szumu.

Oryginalne założenie przyświecało autorowi schematu TIBL (*Typical Instance Based Learning*) (Zhang, 1992). Algorytm ten pozostawia w zbiorze zredukowanym próbki położone blisko środka klastra, a nie brzegowe, jak bywa w przypadku większości innych metod. Obok znacznej odporności na szum i często dużego stopnia redukcji, TIBL ma też wady, do których przede wszystkim trzeba zaliczyć problemy przy skomplikowanych granicach decyzyjnych oraz konieczność modyfikacji procedury, gdy choć jedna klasa w zadaniu zajmuje więcej niż jeden spójny region w przestrzeni.

Dwie metody bardzo agresywnej redukcji przedstawił Cameron-Jones (1995). *ELGrow* (*Encoding Length Grow*) używa skomplikowanej heurystycznej miary opisującej, jak dobrze zbiór zredukowany oddaje powierzchnie decyzyjne zbioru oryginalnego. Próbki ze zbioru są usuwane, jeśli zmniejsza to wartość funkcji kosztu. *ELGrow* jest wrażliwy na kolejność próbek na wejściu. Druga metoda Camerona-Jonesa, *Explore*,

zaczyna od procedury *ELGrow*, a następnie próbuje ulepszyć zbiór przy pomocy szeregu mutacji analogicznych do pierwszego z opisanych algorytmów Skalaka.

Niektóre metody modyfikują zbiór prototypów, zamiast jedynie dokonywania selekcji. Pierwszym algorytmem tego typu była metoda Changa (1974), w której sąsiadujące próbki zostają scalane, jeśli nie narusza to kryterium zgodności. Nowo powstałe prototypy mogą podlegać dalszemu scalaniu, przy czym każdy nowy obiekt jest środkiem ciężkości wszystkich elementów składowych. Jest to *de facto* aglomeratywny algorytm klasteryzacji (Murtagh, 1985). Poważną wadą algorytmu Changa jest jednak bardzo wysoki koszt czasowy uczenia.

Bezdek i in. (1998) zmodyfikowali sposób łączenia prototypów w propozycji Changa, osiągając w swoim algorytmie MCA (*Modified Chang Algorithm*) nieco mniejsze zbiory zredukowane przy jednoczesnym przyspieszeniu uczenia.

Mollineda i in. (2000) rozwinęli ideę Changa, m. in. przez uogólnienie jej do niemal dowolnej metody przyrostowego budowania klastrów. Ubocznym efektem metody jest także znaczne przyspieszenie uczenia. Podobnie jak jego pierwowzory, jest to algorytm zachowujący zgodność zbioru zredukowanego. Algorytm ten umożliwia zredukowanie pełnego zbioru Iris do 9 lub 11 prototypów (w zależności od wersji liczenia odległości między klastrami), co należy uznać za bardzo dobry wynik. Niestety, niełatwo na razie o rzetelną ocenę tego algorytmu, gdyż oprócz zbioru Iris jedynym jeszcze rzeczywistym zbiorem danych, dla którego autorzy podali wyniki, jest zbiór DNA zawierający wyłącznie cechy binarne, a zatem zbiór, który trudno uznać za reprezentatywny dla szerszej klasy praktycznych zadań.

Kilka klasycznych technik formalnie można zaliczyć do redukcji, choć ich głównym celem nie jest zmniejszenie zbioru uczącego, a poprawa jakości, zwłaszcza przy klasyfikacji jednym najbliższym sąsiadem. Mówi się też, iż są to filtry odszumiające i w praktyce opisane niżej algorytmy często wykorzystuje się przed zasadniczym algorytmem redukcji (np. w pracy (Kuncheva, 2001)). Wilson (1972) odrzucał te próbki ze zbioru uczącego, które były źle klasyfikowane przez swoich k najbliższych sąsiadów (ENN — *Edited Nearest Neighbor*); najczęściej przyjmuje się $k = 3$ lub $k = 5$. Tomek (1976b) zaproponował dwie wersje odszumiania: iteracyjne powtarzanie ENN aż do momentu, gdy nic nie można odrzucić (nieraz w literaturze używa się określenia metody: RENN — *Repeated Edited Nearest Neighbor*) oraz „*All k*-NN". Algorytm *All k*-NN każdą próbkę ze zbioru uczącego klasyfikuje przy pomocy $i = 1, 2, ..., k$ sąsiadów. Jeśli dla choć jednej wartości i dana próbka jest źle klasyfiko-

wana, to zostaje oznaczona flagą i na końcu procedury usunięta. Wszystkie z przedstawionych metod oferują nieznaczne zmniejszenie zbioru uczącego, przeważnie nie przekraczające 20%–30%, ale niejednokrotnie poprawiają jakość późniejszej klasyfikacji.

Zamiast usuwania zaszumionych próbek można dokonywać ich reklasyfikacji, tj. nadania nowych etykiet, np. w zgodzie z *k*-NN (analogia do ENN). Celem takiej operacji jest, rzecz jasna, poprawa jakości klasyfikacji, także w schematach połączonych z późniejszą redukcją.

Jóźwik i in. (1995) przedstawili algorytm dzielący hiperpłaszczyznami przestrzeń na regiony, a następnie zastępujący podzbiór zbioru odniesienia z danego regionu pojedynczym prototypem. Ponieważ algorytm ten w pewnym stopniu zachowuje lokalną gęstość zbioru, więc można go używać nie tylko z 1-NN, ale i z *k*-NN. W pracy (Chen i Jóźwik, 1996) wygenerowany zbiór zredukowany został użyty do uczenia sieci neuronowej.

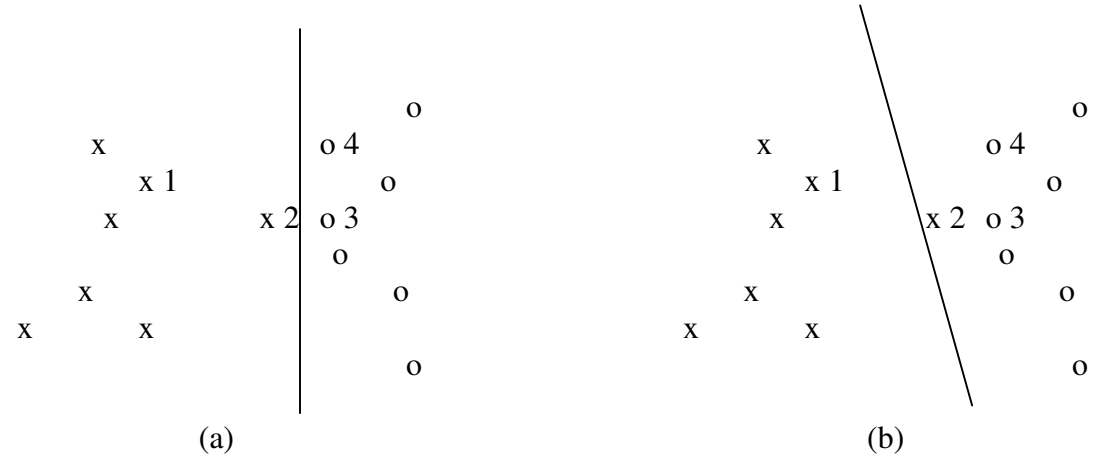
Sánchez i in. (1997a) zaproponowali redukcję przy użyciu grafów Gabriela i RNG (p. Rozdz. 7). W prostszym wariancie ich algorytmu, z oryginalnego zbioru eliminuje się te próbki, których etykieta klasy różni się od etykiety dominującej wśród ich sąsiadów grafowych. Dodatkowo na tak odszumionych zbiorach zastosowano algorytm Tomeka oraz jego odpowiednik używający grafu RNG. Druga faza algorytmów Sáncheza i in. zwykle powoduje pewien spadek jakości predykcji, a zatem jest kompromisem między jakością a stopniem redukcji.

Użycie algorytmów genetycznych do redukcji zbioru odniesienia postulowała Kuncheva (1995). W następnych pracach (Kuncheva i Jain, 1999; Kuncheva, 2001) dokonano tym podejściem jednoczesnej redukcji i selekcji cech.

Wilson i Martinez przedstawili rodzinę algorytmów DROP1..5 (Wilson i Martinez, 1997b; 2000), będących wariacjami nt. algorytmu Gatesa. Nowości polegały zwykle na bardziej subtelnych kryteriach usuwania sąsiadów lub poprzedzeniu algorytmu filtrem odszumiającym zbliżonym do ENN. Jeszcze jedna zaprezentowana metoda redukcji, DEL, łączy podejście autorów ze wspomnianą wyżej heurystyką algorytmu *ELGrow* Camerona-Jonesa. W wymienionych pracach przetestowano proponowane algorytmy na 31 zbiorach UCI i porównano m. in. z CNN i SNN. Oprócz dobrego kompromisu między jakością klasyfikacji a wielkością zbioru uczącego, większość algorytmów Wilsona i Martineza charakteryzuje się też znaczną odpornością na przekłamania etykiet klas zbioru uczącego, czego dowiodły eksperymenty ze sztucznie dodanym szumem.

6.3. O kryterium zgodności

Wiele algorytmów redukcji, zwłaszcza starszych, gwarantuje zgodność zbioru zredukowanego ze zbiorem oryginalnym. Jest to zwykle eleganckie kryterium końca procedury generacji zbioru zredukowanego; można też oczekiwać, że przy dostatecznie dużym zbiorze odniesienia kryterium to zagwarantuje jakość klasyfikacji zredukowanego zbioru porównywalną ze zbiorem oryginalnym. W praktyce jednak dane zbiory odniesienia są notorycznie małe w stosunku do wymiarowości. Powstaje więc pytanie, czy zgodność jest „dobrym" kryterium przy tworzeniu zbioru zredukowanego?



Rys. 6.4. Dwa warianty redukcji zbioru do dwóch obiektów.
Przykład sytuacji, w której wymóg zgodności może prowadzić do przeuczenia.

Rys. 6.4 przedstawia przykład zadania dwudecyzyjnego, dla którego wybrano osobno dwa różne dwuelementowe zbiory zredukowane. W wariancie (a) są to obiekty 2 i 3, zaś w wariancie (b) — obiekty 1 i 4. Którą płaszczyznę rozdzielającą klasy zbioru uczącego należy wybrać? Tylko w wariancie (a) zachowana jest zgodność, jednak wydaje się, iż płaszczyzna (b) lepiej odpowiada hipotetycznemu rozkładowi prawdopodobieństwa. Pojedyncza odstająca od pozostałych (ang. *outlying*) próbka, którą na Rys. 6.4 jest obiekt nr 2, ma znaczący wpływ na wyuczone granice decyzyjne, o ile nałożony jest wymóg zgodności. Przykład ten pokazuje, iż wymóg zgodności zbioru zredukowanego ze zbiorem oryginalnym może prowadzić do gorszej jakości otrzymanego zbioru niż użycie algorytmu, który nie nakłada takiego wymogu. Wyniki testów zamieszczonych w Rozdz. 6.5 potwierdzają tę hipotezę.

6.4. Lokalny wybór zredukowanego zbioru odniesienia

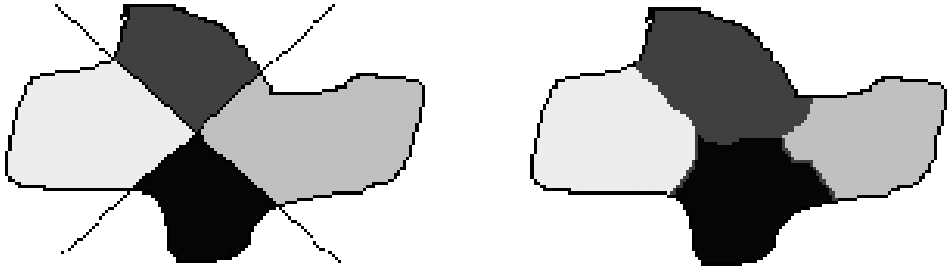
W Rozdz. 3.2.2 omówiono koncepcję lokalnego wyboru klasyfikatora. Jak wspomniano, szybkość klasyfikacji w tej klasie algorytmów jest silnie uzależniona od kryterium wybierającego lokalnie dany klasyfikator. Ponieważ celem poniższych rozważań jest skonstruowanie szybkiego klasyfikatora (dzięki redukcji szybszego niż reguła 1-NN z pełnym zbiorem odniesienia), więc należy wziąć pod uwagę kryteria bardzo szybkie.

Rozważono dwie możliwości szybkiego lokalnego wyboru klasyfikatora dla danej próbki (Grabowski, 2002c; Grabowski i Jóźwik, 2003).

Pierwszy schemat rozdziela zbiór uczący na rozłączne (możliwie skupione) podzbiory, czyli generuje klastry. W procesie uczenia szereg wygenerowanych „globalnie" klasyfikatorów jest testowanych na próbkach należących do poszczególnych pojedynczych klastrów i z każdym klastrem kojarzony jest jeden, najlepszy dla tego klastra, klasyfikator. W czasie klasyfikacji dla testowej próbki wyznaczany jest najbliższy klaster — w sensie najbliższego środka ciężkości klastra — a następnie klasyfikacja dokonywana jest przy pomocy skojarzonego z tym klastrem klasyfikatora. Należy zaznaczyć, że klasteryzacja zbioru odniesienia była już wykorzystywana w schematach klasyfikacji (Jóźwik i Stawska, 1999; Kuncheva, 2000), ale nie łączono jej z redukcją zbioru odniesienia.

Drugi schemat dzieli na rozłączne podzbiory (regiony) nie zbiór uczący, a całą przestrzeń. Ponieważ istotna jest szybkość ustalenia przynależności danej próbki do regionu, więc postanowiono dzielić przestrzeń hiperpłaszczyznami. Przykładowo, trzy hiperpłaszczyzny dzielą przestrzeń na $2^3 = 8$ rozłącznych regionów. Podobnie jak w pierwszym schemacie, z każdym regionem jest skojarzony jeden z nauczonych wcześniej (globalnie) klasyfikatorów: mianowicie ten, który na zbiorze próbek ze zbioru uczącego należących do danego regionu przestrzeni popełnia najmniej błędów. Klasyfikacja polega na znalezieniu regionu przestrzeni, do którego należy dana próbka (przy rozcięciu przestrzeni hiperpłaszczyznami jest to w praktyce bardzo szybkie), a następnie użyciu skojarzonego z wyznaczonym regionem klasyfikatora.

Poglądową ilustrację działania obu metod podziału zbioru przedstawia Rys. 6.5; zbiór odniesienia dzielony jest na 4 regiony.



(a) (b)

Rys. 6.5. Podział zbioru na:
(a) regiony przy pomocy płaszczyzn;
(b) klastry, np. metodą *k* średnich

Poniżej podano szczegóły implementacji obu schematów. W pierwszym schemacie dokonywana jest klasteryzacja zbioru odniesienia znaną metodą *k* średnich (MacQueen, 1967), gdzie zadana liczba klastrów $k = 2^l$. Poniżej przedstawiono formalny opis tego algorytmu.

Uczenie (*L*, *l*)

1. Wygeneruj niezależnie od siebie *L* klasyfikatorów $D_1, ..., D_L$ przy użyciu zbioru odniesienia *S* zawierającego obiekty z przestrzeni *X*.
2. Dokonaj klasteryzacji *S* metodą *k* średnich z zadaną liczbą klastrów $k = 2^l$ otrzymując klastry $C_1, ..., C_k$. Znajdź środki ciężkości otrzymanych klastrów i oznacz je przez $gc_1, ..., gc_k$.
3. Dla każdego klastra C_j estymuj jakość klasyfikacji klasyfikatorów $D_1, ..., D_L$ i oznacz przez $D_{best}(j)$ klasyfikator o najmniejszym błędzie w tym klastrze.
4. Zwróć $gc_1, ..., gc_k$ oraz $D_{best}(1), ..., D_{best}(k)$.

Klasyfikacja próbki *q*:

1. Policz odległości z *q* do wszystkich gc_i. Niech gc_j będzie najbliższym sąsiadem *q* po tym zbiorze.
2. Nadaj etykietę próbce *q* przy pomocy klasyfikatora $D_{best}(j)$.

W schemacie drugim zestaw *l* płaszczyzn dzielących przestrzeń na regiony otrzymywany jest następująco. Ze zbioru uczącego losowanych jest *pc* par próbek — każda para wyznacza jedną hiperpłaszczyznę. Jako pierwsza wybierana jest ta spośród *pc* płaszczyzn, która rozdziela przestrzeń na regiony zawierające możliwie równe ilości próbek ze zbioru uczącego. Następnie w pętli dodawane są kolejne płaszczyzny w taki sposób, aby próbki zbioru uczącego w powstałych regionach były również możliwie równomiernie rozmieszczone. Ściślej mówiąc, użyto kryterium średniokwadratowego: dla każdego aktualnie rozważanego zbioru płaszczyzn liczona jest średnia liczba próbek

w regionie, a następnie wyznaczana jest suma kwadratów odchyleń liczności próbek w każdym regionie od opisanej średniej. Dodatkowo, o ile to możliwe, nie dopuszcza się do powstawania regionów pustych. Płaszczyzna, której dodanie do aktualnego zestawu minimalizuje opisane kryterium, jest przyjmowana w danym kroku. Procedura kończy się po znalezieniu zestawu l płaszczyzn.

Warto zauważyć, że strategia dodawania płaszczyzn jest analogiczna do przyrostowej strategii selekcji cech FSS (Kittler, 1978). Dokładny opis algorytmu przedstawiono poniżej. W obu schematach klasyfikatorami stosowanymi lokalnie są zbiory zredukowane Skalaka (metodą oryginalną i zmodyfikowaną) z regułą 1-NN.

Uczenie (*L*, *l*, *pc*)

1. Wygeneruj niezależnie od siebie L klasyfikatorów D_1, ..., D_L przy użyciu zbioru odniesienia S zawierającego obiekty z przestrzeni X.
2. Wylosuj pc par próbek ze zbioru odniesienia S zawierającego obiekty z przestrzeni X, oznacz zbiór wszystkich par przez P, zaś elementy tego zbioru przez p_i, i=1..pc.
3. Oznacz aktualny zbiór wybranych elementów z P (równoważny zbiorowi płaszczyzn) przez SP. Niech $SP = \varnothing$.
4. Przyjmij i=1.
5. Przyjmij *non_empty*=`FALSE`.
6. Dla każdego j=1..pc:
   6.1. Jeśli $p_j \in SP$, to przejdź do 6.
   6.2. Wyznacz regiony $R(1)$, ..., $R(2^i)$, na które przestrzeń X zostałaby podzielona płaszczyznami ze zbioru $SP \cup p_j$.
   6.3. Wyznacz liczności podzbiorów S należących do regionów $R(1)$, ..., $R(2^i)$; oznacz te liczności odpowiednio przez $rc(1)$, ..., $rc(2^i)$.
   6.4. Jeśli wszystkie $rc(k) > 0$, k=1..2^i, to przyjmij *non_empty*=`TRUE`. W przeciwnym razie przejdź do 6 (tj. weź następne j).
   6.5. Policz średnią liczbę elementów S w regionie: $avg\_r = |S|/2^i$.
   6.6. Policz średnie odchylenie kwadratowe
   $$mse(j) = \sum_{k=1}^{2^i} (rc(k) - avg\_r)^2 .$$
7. Jeśli *non_empty*=`TRUE`, to znajdź j_{opt} minimalizujące $mse(j)$, w przeciwnym razie wykonaj raz jeszcze pętlę 6 z pominięciem kroku 6.4 i podobnie wyznacz j_{opt} minimalizujące $mse(j)$.
8. Przyjmij $SP = SP \cup p_{j_{opt}}$.
9. Jeśli i<l, to zwiększ i o 1 i idź do 5.
10. Otrzymaliśmy 2^l finalnych regionów $R(\cdot)$. Dla każdego regionu $R(j)$ estymuj jakość klasyfikacji klasyfikatorów D_1, ..., D_L i oznacz przez $D_{best}(j)$ klasyfikator o najmniejszym błędzie w tym regionie.
11. Zwróć SP oraz $D_{best}(1)$, ..., $D_{best}(2^l)$.

Klasyfikacja próbki *q*:

1. Ustal region $R(j)$, do którego należy q (wystarczy policzyć $2 \cdot 2^l$ odległości do punktów ze zbioru SP wyznaczających 2^l znalezionych płaszczyzn).
2. Nadaj etykietę próbce q przy pomocy klasyfikatora $D_{best}(j)$.

6.5. Wyniki eksperymentów i dyskusja

W pierwszym eksperymencie przeprowadzono porównanie kilku znanych z literatury algorytmów redukcji na parach klas zbioru FerritesOld. Jedynym algorytmem autora rozprawy użytym w tym teście był opisany wyżej SNN-CR (modyfikacja algorytmu selektywnego). Zbiór odniesienia obejmuje 5 klas i liczy 1000 elementów posiadających 12 cech. Zbiór testowy zawiera 1885 elementów. W obu zbiorach: uczącym i testowym, klasy są równoliczne.

Pięć klas w zbiorze oznacza dziesięć osobnych binarnych klasyfikatorów działających ze zbiorami zredukowanymi. Decyzja globalna w zadaniu wielodecyzyjnym mogłaby zatem zostać otrzymana w wyniku głosowania klasyfikatorów składowych w schemacie Jóźwik–Vernazza. Warto zauważyć, że zgodność zbioru zredukowanego dla każdej pary klas pociąga za sobą „globalną" zgodność klasyfikatora wielodecyzyjnego. Jeśli bowiem próbka ze zbioru oryginalnego należąca do dowolnej klasy i jest poprawnie rozpoznawana przez wszystkie i klasyfikatory dwudecyzyjne zawierające i-tą klasę, to żadna inna klasa w głosowaniu nie może otrzymać więcej niż $i-1$ głosów. W testach ograniczono się jednak do podania wielkości zbiorów zredukowanych i jakości klasyfikacji tylko dla wszystkich par klas osobno. Wszystkie zbiory zostały znormalizowane, użyto metryki euklidesowej. Przed redukcją została przeprowadzona selekcja cech, metodą kolejnego dołączania (FSS). Zamieszczone wyniki pochodzą z pracy (Grabowski, 2000a).

Tab. 6.2 informuje o wielkościach zbiorów zredukowanych i estymowanych błędach klasyfikacji dla każdej pary klas. W drugiej kolumnie tabeli podana jest liczba wyselekcjonowanych cech, zaś w trzeciej — błąd klasyfikacji estymowany przy użyciu oryginalnego zbioru próbek danych dwóch klas jako zbioru odniesienia (w każdym przypadku były to zbiory 400-elementowe).

Tabela 6.2: Porównanie algorytmów redukcji zbioru odniesienia
na parach klas zbioru FerritesOld

para klas	liczba cech	zb. oryg.	Hart		G–K		SNN		SNN-CR		Tomek	
		błąd [%]	zb. zred.	błąd [%]	zb. zred.	błąd [%]	zb. zred.	błąd [%]	zb. zred.	błąd [%]	zb. zred.	błąd [%]
1 / 2	1	0,00	4	0,00	2	0,00	2	0,00	2	0,00	2	0,00
1 / 3	12	9,02	72	10,21	66	9,42	76	9,42	74	9,55	172	9,02
1 / 4	7	9,95	78	12,73	63	11,94	64	13,40	67	13,26	185	10,48
1 / 5	2	1,19	13	1,06	10	1,46	10	1,06	10	1,06	17	1,19
2 / 3	1	0,00	4	0,13	2	0,00	2	0,00	2	0,00	4	0,00
2 / 4	1	0,13	3	0,13	2	0,13	2	0,80	2	0,13	2	0,13
2 / 5	8	4,11	50	6,10	39	6,50	43	5,57	43	5,57	106	4,24
3 / 4	5	30,50	175	34,06	167	35,41	169	34,62	168	34,09	331	30,77
3 / 5	3	2,52	21	2,92	14	2,92	18	3,05	18	3,05	36	2,52
4 / 5	2	1,06	9	1,46	9	1,46	7	1,72	7	1,72	13	1,06

Analiza wyników Tab. 6.2 pozwala na dokonanie kilku obserwacji:

- w przypadkach najłatwiejszych (klasy liniowo rozdzielne lub prawie rozdzielne) wszystkie testowane algorytmy uzyskały wyniki optymalne (błąd zerowy lub bardzo niski), zarazem przy bardzo skutecznej redukcji;
- żaden z algorytmów redukcji w żadnym przypadku nie oferował błędu klasyfikacji niższego niż otrzymywany z użyciem pełnego zbioru odniesienia; tym niemniej błąd dla zbioru Tomeka był często identyczny lub nieznacznie wyższy niż dla pełnego zbioru. Ceną wysokiej jakości predykcji przy algorytmie Tomeka był stosunkowo niski stopień redukcji (zbiory często dwa i więcej razy większe od zbiorów np. Gowdy–Krishny);
- zaproponowana modyfikacja algorytmu selektywnego, tj. SNN-CR, prowadziła do poprawy jakości w trzech przypadkach, pogorszenia w jednym, natomiast w sześciu przypadkach nie miała znaczenia. Liczności zbiorów zredukowanych były średnio takie same, zaś wahania dla poszczególnych przypadków — zerowe lub minimalne;
- stopień redukcji algorytmów Harta, G–K, SNN i SNN-CR był często bardzo zbliżony; średnio największy stopień redukcji oferował algorytm G–K, nieraz jednak (pary klas: 1 / 5 i 2 / 5) za cenę niższej jakości w stosunku do algorytmów konkurencyjnych;
- przy wszystkich algorytmach widać silny związek wielkości zbioru zredukowanego z błędem (stopniem zaszumienia danej pary klas): im większy stopień nakładania się klas, tym większy zbiór zredukowany.

W kolejnych testach uwzględniono tylko dwa z algorytmów redukcji z pierwszego eksperymentu, a mianowicie algorytmy Harta (CNN) i G–K. Przemawia za tym wyborem ich szybkość oraz stosunkowo dobry stopień redukcji (w stosunku do algorytmu Tomeka); ponadto CNN jest bodaj najczęściej cytowanym i używanym do porównań algorytmem w literaturze. Dodatkowo jednak przetestowano algorytm Skalaka w dwóch wersjach (oryginalnej i zaproponowanej w Rozdz. 6.2) oraz kilka wersji schematu lokalnego wyboru zbioru zredukowanego.

Eksperymenty przeprowadzono na dwóch dużych zbiorach: Ferrites i Remotes. Nie stosowano selekcji cech.

Wyniki w Tab. 6.3 to średnie z eksperymentów na 10 partycjach zbioru Ferrites. W przypadku stosowania redukcji Harta, G–K oraz reguły 1-NN bez jakiejkolwiek redukcji (pierwsze trzy wiersze w tabeli) są to średnie z 10 wynikowych błędów oraz liczności zbioru zredukowanego. Pozostałe algorytmy mają naturę probabilistyczną, a zatem wydawało się słuszne przeprowadzenie większej liczby eksperymentów. Konkretnie, dla każdej z 10 partycji uruchomiono dany algorytm pięciokrotnie — co oznacza, że kolejne wiersze w tabeli przechowują średnie z 50 „pojedynczych" eksperymentów.

Tabela 6.3: Liczności zbiorów zredukowanych i estymowana jakość klasyfikacji wybranych schematów z redukcją zbioru odniesienia i regułą decyzyjną 1-NN na zbiorze Ferrites

nr	algorytm	liczność	błąd [%]
1	1-NN bez redukcji	1400	11,60
2	Hart (CNN)	388.0	14,01
3	Gowda–Krishna	358.1	14,27
4	Skalak1	30	13,30
5	Skalak2	30	13,00
6	Skalak1 + klast.	30–16=14	12,76
7	Skalak2 + klast.	30–16=14	12,41
8	Skalak1 + PP	30–8=22	12,67
9	Skalak2 + PP	30–8=22	12,50
10	Skalak1	100	13,04
11	Skalak2	100	13,01
12	Skalak1 + klast.	100–16=84	12,93
13	Skalak2 + klast.	100–16=84	12,92
14	Skalak1 + PP	100–8=92	12,86
15	Skalak2 + PP	100–8=92	12,82

Wiersz pierwszy Tab. 6.3 przedstawia wyniki działania reguły jednego najbliższego sąsiada bez stosowania redukcji. Wiersze 2–3 ukazują efekty działania tradycyjnych algorytmów redukcji Harta i Gowdy–Krishny. W wierszach 4–9 zaprezentowano wyniki algorytmów bazujących na redukcji Skalaka, gdzie wielkość zbioru zreduko-

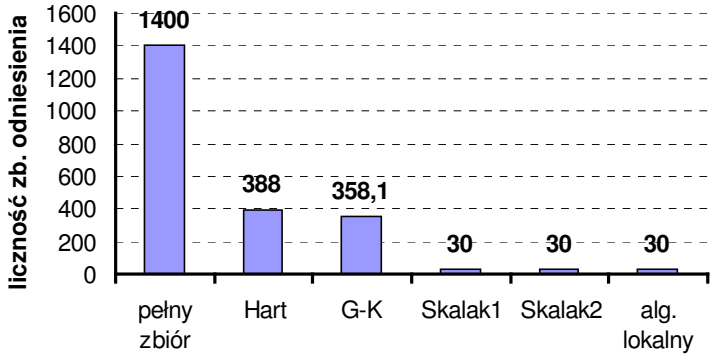
wanego wynosi 30 próbek — lub specyfika algorytmu (tj. lokalny wybór zbioru zredukowanego) wymaga użycia jeszcze mniejszego zbioru odniesienia, aby szybkość klasyfikacji była taka, jak w przypadku tradycyjnej redukcji zbioru uczącego do 30 elementów. *Skalak1* oznacza oryginalny algorytm Skalaka, natomiast *Skalak2* to modyfikacja autora rozprawy tworząca sztuczne prototypy. Wiersze 6–7 to schemat z klasteryzacją metodą *k* średnich odpowiednio na bazie redukcji *Skalak1* i *Skalak2*. Liczba klastrów została ustalona na 16, a zatem wielkość każdego zbioru zredukowanego w tych schematach musi być zmniejszona do 14 (tj. 30–16) próbek. Wiersze 8 i 9 wreszcie, to schemat z partycjonowaniem przestrzeni (PP) na bazie odpowiednio *Skalak1* i *Skalak2*. Przy 16 regionach ustalenie przynależności testowej próbki do regionu wymaga znalezienia orientacji próbki względem czterech płaszczyzn, tj. policzenia ośmiu odległości — stąd składowe zbiory zredukowane liczą po 22 (=30–8) próbki.

Ponieważ problem dogodnej wielkości zbioru zredukowanego jest wciąż otwarty, postanowiono przetestować powyższe algorytmy probabilistyczne również w przypadku czasu klasyfikacji równoważnego klasycznej redukcji zbioru tym razem do 100 elementów. Wyniki tej grupy testów zawarte są w wierszach 10–15.

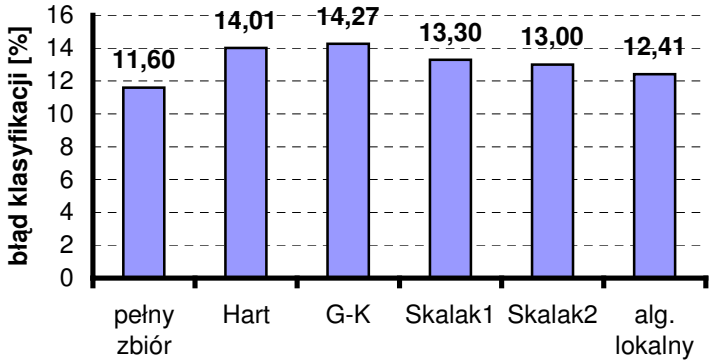
Należy jeszcze podać pozostałe parametry testowanych algorytmów. Liczba mutacji m_1 we wszystkich algorytmach na bazie *Skalak1* wynosiła 1000; taka sama była też wartość m_2 we wszystkich schematach wykorzystujących procedurę *Skalak2*. Liczba klasyfikatorów L we wszystkich schematach z lokalnym wyborem klasyfikatora wynosiła 25. W algorytmach z partycjonowaniem przestrzeni przyjęto $pc = 1000$ kandydujących płaszczyzn.

Najważniejsze wyniki z Tab. 6.3 zaprezentowano również w formie graficznej na Rys. 6.6 i 6.7. Kolejne słupki dotyczą klasyfikacji regułą 1-NN z:

1. pełnym zbiorem odniesienia;
2. zbiorem zredukowanym algorytmem Harta;
3. zbiorem zredukowanym algorytmem Gowdy–Krishny;
4. zbiorem zredukowanym oryginalnym algorytmem Skalaka;
5. zbiorem zredukowanym zmodyfikowanym algorytmem Skalaka;
6. zespołem zbiorów zredukowanych zmodyfikowanym algorytmem Skalaka z lokalnym wyborem pojedynczego zbioru przy użyciu kryterium opartego na klasteryzacji oryginalnego zbioru odnieniesienia.



Rys. 6.6. Liczba odległości obliczanych przy klasyfikacji pojedynczej próbki dla wybranych algorytmów opartych na regule 1-NN. Eksperymenty na zbiorze Ferrites.



Rys. 6.7. Błąd klasyfikacji wybranych algorytmów opartych na regule 1-NN na zbiorze Ferrites

Analiza Tab. 6.3 pozwala na dokonaniu kilku spostrzeżeń.

- Pod względem jakości klasyfikacji bezkonkurencyjny okazał się... brak redukcji. Szybkość klasyfikacji w tym przypadku — kilkanaście lub kilkadziesiąt razy mniejsza niż dla większości pozostałych metod — zniechęca jednak do tego podejścia w sytuacji, gdy zbiór odniesienia jest bardzo duży.
- Algorytmy: Harta i G–K zdecydowanie przegrywają z algorytmami natury stochastycznej, tak pod względem stopnia redukcji, jak i jakości klasyfikacji. Godny odnotowania jest większy błąd algorytmu G–K niż metody Harta, mimo iż algorytm Gowdy i Krishny był w intencji jego twórców algorytmu Harta usprawnieniem. Różnica nie jest jednak duża i nie można wyciągać przedwczesnych uogólnień.

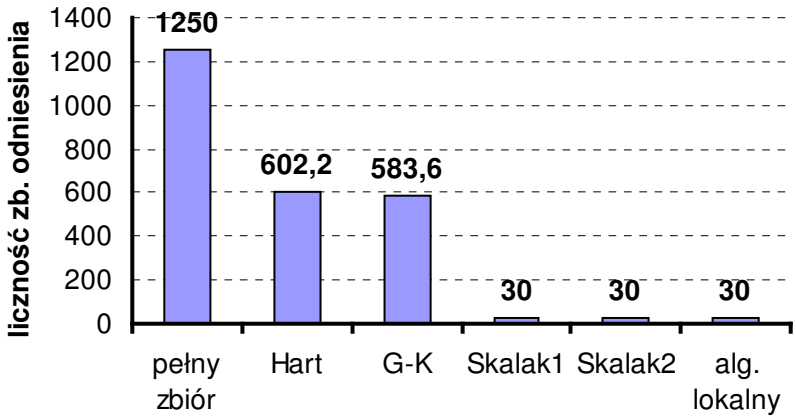
- Niepowodzenie algorytmów Harta i G–K stawia też pod znakiem zapytania stosowność używania kryterium zgodności zbioru odniesienia ze zbiorem oryginalnym przy konstrukcji algorytmów redukcji. Pozostałe testowane algorytmy zgodności nie gwarantowały (i w praktyce od zgodności były odległe).
- Zaproponowana modyfikacja algorytmu Skalaka wiedzie do pewnej poprawy jakości, o ile zadana redukcja jest bardzo ostra. Przy łagodniejszej redukcji różnice błędów obu procedur (także w schematach złożonych) były nieznaczące.
- Lokalny wybór zredukowanego zbioru odniesienia prowadzi do poprawy jakości klasyfikacji w porównaniu z algorytmami redukcji globalnej przy tej samej, zadanej, szybkości klasyfikacji. Zysk jakościowy większy był w przypadku redukcji bardziej radykalnej.
- W schematach z lokalnym wyborem zredukowanego zbioru odniesienia mniejszy błąd otrzymano przy bardziej zdecydowanej redukcji. Fakt ten jest optymistyczny: sugeruje, iż przynajmniej w niektórych praktycznych sytuacjach granice decyzyjne między zbiorami są stosunkowo proste, zaś bardziej złożone klasyfikatory mogą być bardziej narażone na przeuczenie.
- Na podstawie uzyskanych wyników nie można stwierdzić, które z podejść do problemu generacji klasyfikatorów i lokalnego wyboru klasyfikatora: oparte na klasteryzacji zbioru odniesienia czy oparte na partycjonowaniu przestrzeni, jest lepsze.

Wyniki drugiego eksperymentu, dotyczącego zbiorów Remotes, zamieszczone są w Tabeli 6.4. Eksperymenty wykonano z trzema wersjami zbioru (Remotes100, Remotes150, Remotes250) różniącymi się licznościami zbioru uczącego (od zbioru zawierającego po 100 próbek każdej klasy do zbioru przechowującego po 250 próbek na klasę). Zbiory testowe zawierały wszystkie pozostałe próbki zbioru Remotes.

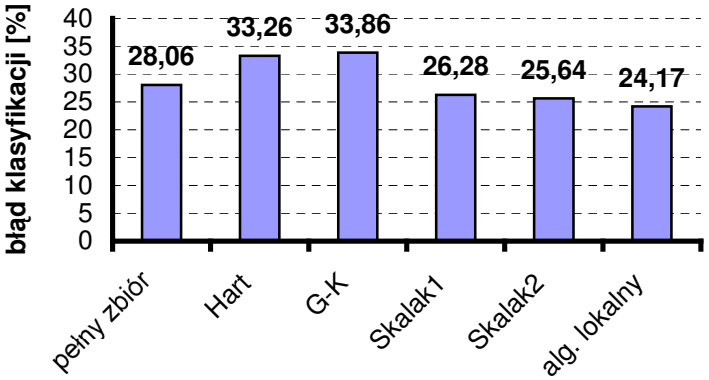
Ponadto na Rys. 6.8 i 6.9 zwizualizowano szybkości i błędy wybranych klasyfikatorów (tych samych, których wyniki dla zbioru Ferrites przedstawiono na Rys. 6.6 i 6.7) na zbiorze Remotes250.

Tabela 6.4: Liczności zbiorów zredukowanych i estymowana jakość klasyfikacji wybranych schematów z redukcją zbioru odniesienia i regułą decyzyjną 1-NN na zbiorach Remotes (po 100, 150 i 250 próbek na klasę w zbiorze uczącym)

nr	algorytm	Remotes100		Remotes150		Remotes250	
		liczność	błąd [%]	liczność	błąd [%]	liczność	błąd [%]
1	1-NN bez redukcji	500	30,85	750	28,85	1250	28,06
2	Hart (CNN)	266,2	36,26	379,7	34,09	602,2	33,26
3	Gowda–Krishna	258,1	36,52	365,5	34,46	583,6	33,86
4	Skalak1	30	29,18	30	28,05	30	26,28
5	Skalak2	30	29,00	30	27,31	30	25,64
6	Skalak1 + klast.	30–16=14	28,42	30–16=14	26,65	30–16=14	25,49
7	Skalak2 + klast.	30–16=14	27,68	30–16=14	25,91	30–16=14	24,17
8	Skalak1 + PP	30–8=22	28,54	30–8=22	26,98	30–8=22	25,45
9	Skalak2 + PP	30–8=22	28,27	30–8=22	26,54	30–8=22	24,72
10	Skalak1	100	30,75	100	27,74	100	25,90
11	Skalak2	100	30,53	100	27,99	100	26,02
12	Skalak1 + klast.	100–16=84	31,28	100–16=84	28,25	100–16=84	25,69
13	Skalak2 + klast.	100–16=84	30,66	100–16=84	27,93	100–16=84	25,51
14	Skalak1 + PP	100–8=92	31,15	100–8=92	28,26	100–8=92	25,73
15	Skalak2 + PP	100–8=92	30,84	100–8=92	28,18	100–8=92	25,67



Rys. 6.8. Liczba odległości obliczanych przy klasyfikacji pojedynczej próbki dla wybranych algorytmów opartych na regule 1-NN. Eksperymenty na zbiorze Remotes250.



Rys. 6.9. Błąd klasyfikacji wybranych algorytmów opartych na regule 1-NN na zbiorze Remotes250

Porównanie Tabel 6.3 i 6.4 wskazuje, iż zbiory Ferrites i Remotes mają znacząco odmienne charakterystyki, choć naturalnie istnieją podobieństwa. W szczególności, dla ostatniego eksperymentu nasuwają się następujące spostrzeżenia:

- zaszumienie zbiorów oryginalnych jest na tyle duże, iż reguła 1-NN z pełnym zbiorem odniesienia zwraca wyższy błąd klasyfikacji niż oferowany przez wiele schematów z redukcją;
- do największych błędów predykcji prowadzi zastosowanie algorytmów Harta i G–K; podobnie jak w poprzednim eksperymencie, zbiór G–K jest mniejszy od zbioru Harta, ale jakość predykcji zbioru G–K jest niższa; ponownie stawia to w niekorzystnym świetle zarówno kryterium zgodności, jak i metodę „walki" z niedostatkami algorytmu Harta, wybraną przez Gowdę i Krishnę;
- modyfikacja algorytmu Skalaka, tj. procedura *Skalak2*, przynosi poprawę jakości przy małych zbiorach zredukowanych (zjawisko podobne do zaobserwowanego na Ferrytach), przy czym zysk jakościowy jest wyższy dla liczniejszych zbiorów odniesienia (Remotes150 i Remotes250), co oznacza prawdopodobnie, iż redukcja do 30 elementów przy bardziej „gęstych" zbiorach odniesienia jest zbyt silna (lub liczba mutacji m_1 zbyt mała) i poszerzenie przestrzeni poszukiwań poprzez przesunięcia próbek charakterystyczne dla *Skalak2* odnosi korzystny skutek;
- analiza wierszy 4–9 potwierdza, iż lokalny wybór zbioru zredukowanego prowadzi do zwiększenia predykcji, jeśli zbiory zredukowane są małe; w przypadku redukcji do 100 próbek (lub przy ekwiwalentnej szybkości klasyfikacji) lokalny wybór zbioru praktycznie nie poprawia jakości predykcji, a nawet może prowadzić do niewielkiego jej pogorszenia;
- wydaje się, że klasteryzacja jest nieco lepszą metodą dekompozycji zbioru dla schematów z lokalnym wyborem klasyfikatora niż partycjonowanie przestrzeni hiperpłaszczyznami, jednak różnice między obu metodami są bardzo małe.

Podsumowując wyniki z Tabel 6.2, 6.3 i 6.4, należy stwierdzić rzeczy następujące.

- Klasyczne algorytmy redukcji: Harta, Gowdy–Krishny i algorytm selektywny nie umożliwiają uzyskania jakości klasyfikacji porównywalnej z jakością reguły 1-NN na pełnym zbiorze odniesienia. Z metod klasycznych, jedynie algorytm

Tomeka osiąga stosunkowo dobrą jakość, ale za cenę niskiej redukcji (na zbiorze Ferrites algorytm Tomeka, przy metryce euklidesowej, a więc przy nieco innej metodologii testu niż użytej dla Tab. 6.3, osiągnął średni błąd 12,79% przy średniej redukcji do 1036 próbek). Nieefektywność przetestowanych metod stawia pod znakiem zapytania zasadność nakładania wymogu zgodności zbioru zredukowanego ze zbiorem oryginalnym.

- Kwestią wartą osobnego zbadania pozostaje wpływ filtrów odszumiających (np. ENN), które mogą być użyte przed zasadniczym algorytmem redukcji.
- Algorytm Skalaka, jako procedura nie kierująca się kryterium zgodności i redukująca zbiór odniesienia do wielkości zadanej przez użytkownika, osiągał znacząco wyższą jakość klasyfikacji niż algorytmy Harta i G–K, przy co najmniej kilkakrotnie wyższym stopniu redukcji. Interesującym zjawiskiem jest fakt, że — przy tym samym algorytmie redukcji — jakość predykcji przy mniejszym zbiorze zredukowanym może być wyższa niż przy zbiorze większym.
- Zaproponowana modyfikacja algorytmu Skalaka, przesuwająca losowo próbki wygenerowanego zbioru zredukowanego, prowadzi nieraz do wyższej jakości klasyfikacji (a niemal nigdy, jak wynika z prezentowanych tabel, nie szkodzi); efekt taki wydaje się mieć miejsce w przypadkach, gdy zadana wielkość zbioru zredukowanego jest mała w stosunku do złożoności problemu.
- Zaproponowane podejście wybierające odpowiedni zbiór zredukowany dla danej próbki wejściowej dobrze współdziała z algorytmem Skalaka (w wersji oryginalnej i zmodyfikowanej), zaś otrzymywana jakość klasyfikacji przewyższa jakość oferowaną przez „pojedyncze" zbiory zredukowane, zwłaszcza przy agresywnej redukcji. Godne podkreślenia jest szybkie kryterium wyboru zbioru zredukowanego, dzięki czemu możliwe jest porównanie schematu z algorytmem Skalaka przy tej samej szybkości klasyfikacji.
- Z zaproponowanych dwóch podejść do zagadnienia dekompozycji zbioru odniesienia, klasteryzacja metodą *k* średnich wydaje się prowadzić do nieco lepszych rezultatów.
- Kwestią otwartą pozostaje wybór wielkości zbioru zredukowanego oraz wybór liczby mutacji w procedurach *Skalak1* i *Skalak2*. Szczególnie istotna jest odpowiedź na pierwsze pytanie i będzie ona przedmiotem dalszego zainteresowania autora.

Rozdział VII
Klasyfikatory oparte na koncepcji symetrycznego sąsiedztwa

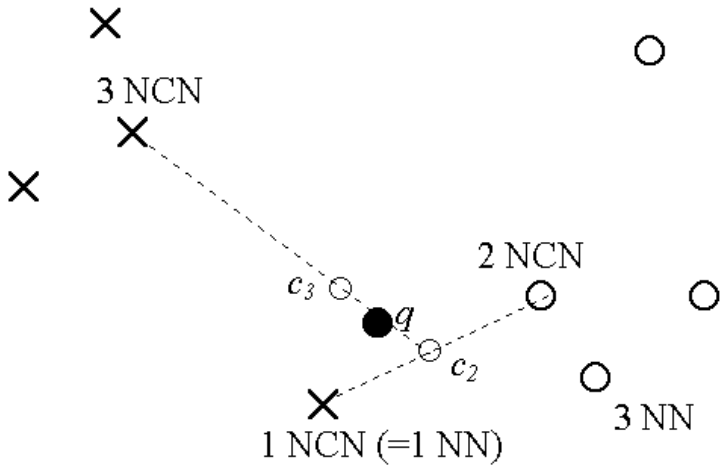
Pojęcie sąsiedztwa (ang. *neighborhood*) używane jest nie tylko w klasycznym rozpoznawaniu obrazów, ale i w takich — często pokrewnych — dziedzinach, jak „widzenie" maszynowe (ang. *computer vision*), geometria wyliczeniowa (ang. *computational geometry*), przetwarzanie obrazu (ang. *image processing*) i inne. Ogólnie biorąc, sąsiadami próbki q w d-wymiarowej przestrzeni są obiekty położone blisko q. Sama bliskość (ang. *proximity*) nie musi być jedyną pożądaną właściwością sąsiedztwa. Wydaje się, iż korzystnie jest, gdy sąsiedzi umiejscowieni są w przestrzeni cech możliwie symetrycznie wokół obiektu q. Reguła k najbliższych sąsiadów ignoruje ów drugi aspekt sąsiedztwa. Jak wiadomo, asymptotyczna optymalność reguły k-NN nie gwarantuje odpowiednio wysokiej jakości klasyfikacji w przypadkach rzeczywistych (skończonych). Zatem próby poszukiwań nowych klasyfikatorów, także opartych o kryterium „symetryczności" sąsiedztwa, są uzasadnione.

W niniejszym rozdziale zaprezentowano kilka klasyfikatorów działających z wykorzystaniem koncepcji symetrycznego sąsiedztwa (ang. *surrounding neighborhood*), zaczynając od rozwiązań podawanych w literaturze i przechodząc stopniowo do własnych idei autora. Wprowadzono regułę decyzyjną *k Near Surrounding Neighbors* (k-NSN) (Grabowski, 2002d) będącą rozwinięciem idei klasyfikatora „k scentrowanych sąsiadów" (k-NCN) i optymalizującą jednocześnie oba wykorzystywane przez k-NCN kryteria, opisano schematy *voting* k-NN (Grabowski, 2002b), *voting* k-NCN i *voting* k-NSN (Grabowski, i in., 2003; Grabowski, 2003a) wykorzystujące opracowaną przez autora rozprawy koncepcję głosowania homogenicznych (tj. o jednakowej strukturze) klasyfikatorów z różnymi wartościami parametru k, a także zmodyfikowano klasyfikatory grafowe GGN i RNGN poprzez redukcję ich sąsiedztwa przy użyciu kryterium środka ciężkości (Grabowski, 2001b). Zaprezentowano również rodzinę klasyfikatorów kaskadowych stosujących w pierwszej fazie równoległą sieć klasyfikatorów homogenicznych (np. typu k-NN), zaś w fazie drugiej dokonujących predykcji na bazie symetrycznego sąsiedztwa (Grabowski, 2003a). Wszystkie opisane klasyfikatory zostały przetestowane na wielu zbiorach. Ponadto przetestowano

schematy dekompozycji zadań wielodecyzyjnych: Jóźwik–Vernazza i Moreira–Mayoraz, dla kilku opisywanych klasyfikatorów, czego nawet dla znanej od kilku lat koncepcji *k*-NCN brakowało w literaturze.

7.1. Reguła decyzyjna *k* scentrowanych sąsiadów (*k*-NCN)

Chaudhuri (1996) podał interesującą koncepcję, rozwiniętą później (Sánchez i in., 1997b) do reguły decyzyjnej, skupiającą się na sąsiadach położonych nie tylko blisko danej próbki, ale także rozłożonych możliwie jednorodnie wokół niej. Regułę decyzyjną, o której mowa, określa się w niniejszej pracy mianem reguły *k scentrowanych sąsiadów*[12] (ang. *k Nearest Centroid Neighbors*) lub wymiennie oryginalnym skrótem *k*-NCN. *K* scentrowanych sąsiadów próbki *q* otrzymuje się w sposób następujący:
– pierwszym sąsiadem próbki *q* jest jej najbliższy sąsiad, oznaczmy go przez n_1;
– *i*-tym sąsiadem, n_i, $i \geq 2$, jest próbka, dla której centroid (tj. środek ciężkości) zbioru złożonego z n_i i wszystkich uprzednio wybranych sąsiadów n_1, ..., n_{i-1} jest położony możliwie najbliżej *q*.



Rys. 7.1. Reguła *k*-NCN, *k*=3. Próbka *q* zostanie przypisana do klasy „krzyżyków".

Nietrudno zauważyć, że wybór pierwszego scentrowanego sąsiada nie jest arbitralny, gdyż środek ciężkości zbioru zawierającego tylko jednego sąsiada również jest zminimalizowany (zatem definicję można zapisać zwięźlej, tracąc nieco na

[12] Tłumaczenie terminu pochodzi od autora rozprawy.

przejrzystości). Przykład działania reguły *k*-NCN dla $k = 3$ jest przedstawiony na Rys. 7.1.

Z definicji sąsiadów NCN, jak będziemy wymiennie nazywać scentrowanych sąsiadów wynika, że pod uwagę brany jest również przestrzenny układ rozważanego podzbioru. Z drugiej strony, iteracyjny sposób, w jaki znajdowani są kolejni sąsiedzi gwarantuje ich bliskość względem próbki *q*.

Eksperymenty przeprowadzone przez Sáncheza i in. (Sánchez i in., 1997b, 1998) zarówno na zbiorach rzeczywistych, jak i sztucznych, potwierdziły atrakcyjność reguły *k*-NCN, szczególnie w zastosowaniach, gdzie jakość klasyfikacji jest ważniejsza od szybkości. Reguła *k*-NCN zazwyczaj oferuje wyższą jakość klasyfikacji niż *k*-NN.

W eksperymentach autora rozprawy (Jóźwik i Grabowski, 2003) reguła *k*-NCN została także z sukcesem wykorzystana do odszumiania zbioru odniesienia w duchu idei Wilsona (1972) używanej tradycyjnie z regułą *k*-NN.

Podobnie jak przy *k*-NN, także dla reguły *k*-NCN liczba sąsiadów (tj. parametr *k*) musi być estymowana przy użyciu zbioru uczącego, np. metodą minus jednego elementu.

Idea *k*-NCN z lokalnie wybieranym *k*

Zachłanny (ang. *greedy*) charakter procedury otrzymywania kolejnych sąsiadów w *k*-NCN zachęca do prób poszerzenia przestrzeni poszukiwań sąsiadów, stale jednak z uwzględnieniem kryterium środka ciężkości układu. Wydaje się to jednak trudne z uwagi na kryterium bliskości. Idea przedstawiona w niniejszym punkcie jest bardzo prosta: dla danej próbki *q* znajdowanych jest jej *k* scentrowanych sąsiadów, a następnie wybierana taka liczba $k(q)$, $k(q) \le k$, pierwszych sąsiadów n_1, ..., $n_{k(q)}$, aby odległość od centroidu $c_{k(q)}$ do *q* była zminimalizowana po wszystkich odległościach z centroidów c_j, $j = 1, ..., k$. Innymi słowy, kryterium środka ciężkości zostaje zminimalizowane lokalnie, co w niewielkim stopniu modyfikuje oryginalną procedurę. Wartość *k*, będąca w tym wariancie maksymalną liczbą sąsiadów, musi być wyznaczona wcześniej (globalnie, *off-line*).

Niestety, opisana reguła decyzyjna zawodzi na większości zbiorów danych (z wyjątkiem zbioru Ferrites, gdzie średni błąd lokalnej *k*-NCN wyniósł 10,17%, tj. praktycznie tyle samo, co dla oryginalnej reguły *k*-NCN). Podano ją jedynie dlatego, iż idea ta (tj. lokalny wybór liczby sąsiadów przy użyciu kryterium środka ciężkości

układu) została jednak z pewnym sukcesem wykorzystana do konstrukcji nowych klasyfikatorów grafowych, opisanych w Rozdz. 7.3.

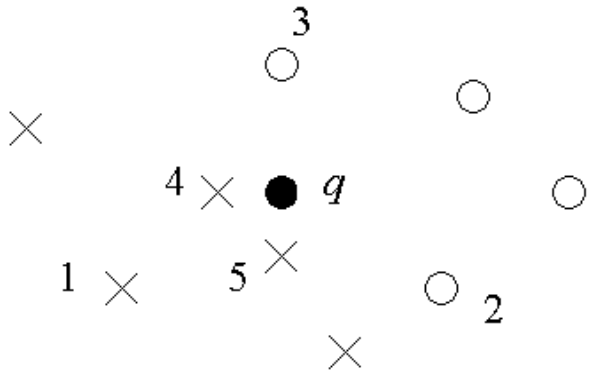
Równoległa reguła *k*-NCN

Zaproponowano w tym punkcie naturalne modyfikacje reguły *k*-NCN, będące analogią do schematów przedstawionych w pracach: (Jóźwik i Vernazza, 1988) dla reguły *k*-NN oraz (Moreira i Mayoraz, 1998) dla drzew decyzyjnych. Zamiast zmagać się z problemem *c*-decyzyjnym, dekomponuje się go na sieć podzadań dwudecyzyjnych. Liczba podzadań wynosi $c \cdot (c-1)/2$ w schemacie Jóźwika–Vernazzy (J–V) lub $c \cdot (c-1)$ w wersji Moreiry–Mayoraza (M–M). Oba schematy dekompozycyjne opisano w Rozdz. 3.2.

Praktyczną trudnością związaną z regułą *k*-NCN (zwłaszcza w wersji z dekompozycją), są wysokie koszta selekcji cech w modelu „wrapper". Jeśli zatem schemat bazujący na *k*-NCN ma używać selekcji cech, to można zasugerować estymację jakości zestawów cech przy użyciu reguły *k*-NN.

7.2. Proponowana reguła *k Near Surrounding Neighbors* (*k*-NSN)

W klasyfikatorze *k*-NCN oba używane kryteria — bliskość i symetryczność sąsiedztwa — są istotne. Rezygnacja z wymogu bliskości prowadzi na manowce, co obrazuje Rys. 7.2. Gdyby decydowało jedynie kryterium środka ciężkości, trzema wybranymi sąsiadami *q* byłyby próbki 3, 2 i 1, co jest oczywiście niekorzystne. W rzeczywistości, reguła 3-NCN wybierze jako sąsiadów próbki 4, 5 i 3. Na szczęście podobna sytuacja jest niemożliwa przy *k*-NCN ze względu na iteracyjne dołączanie sąsiadów.



Rys. 7.2. Przykład obrazujący potrzebę wymogu bliskości w definicji sąsiedztwa NCN

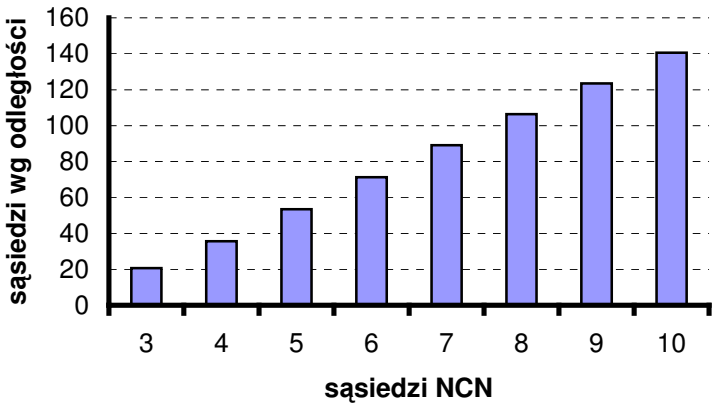
Skalak (1997) zauważył, że klasyfikator *k*-NN może być postrzegany jako równoległa sieć *k* komponentów z głosowaniem większościowym: jeden komponent to klasyfikator typu *pierwszy najbliższy sąsiad* (czyli po prostu 1-NN), następny to *drugi najbliższy sąsiad* itd. Podobnie można postrzegać klasyfikator *k*-NCN. Jak wiemy, jakość predykcji zespołu klasyfikatorów zależy od jakości komponentów oraz stopnia ich korelacji między sobą. Ponieważ dowolny *i*-ty scentrowany sąsiad próbki *q* leży przeważnie dalej od *q* niż *i*-ty sąsiad *q* w sensie odległości (czyli używany w *k*-NN), więc z pewnością klasyfikator typu *i*-ty scentrowany sąsiad będzie średnio gorszym predyktorem niż *i*-ty najbliższy sąsiad.[13] Oznacza to, iż *k*-NCN jest siecią klasyfikatorów średnio gorszych od klasyfikatorów składowych używanych w *k*-NN. Tym niemniej, jak wynika z późniejszych testów (Rozdz. 7.6), *k*-NCN osiąga przeważnie niższe błędy predykcji. Wyjaśnieniem zjawiska musi być mniejsza korelacja klasyfikatorów składowych przy *k*-NCN. W przyszłości autor pracy planuje przeprowadzenie ilościowej analizy tego fenomenu.

Nietrudno jednak sprawdzić, jak położeni są scentrowani sąsiedzi w sensie odległości do testowej próbki. Miarą takiego rozkładu może być liczba „konwencjonalnych" (tj. wyznaczanych wg odległości) sąsiadów w kuli rozpiętej przez najdalszego spośród *k* scentrowanych sąsiadów. Na Rys. 7.3 przedstawiono liczności zbiorów najbliższych sąsiadów w kuli sąsiedztwa *k*-NCN dla pojedynczej partycji zbioru Ferrites. Oś odciętych oznacza liczbę sąsiadów scentrowanych *k* (od 3 do 10), zaś oś rzędnych — średnią liczbę sąsiadów opartych na odległości w indukowanych scentrowanym sąsiedztwem kulach.

Liczności zbiorów najbliższych sąsiadów na Rys. 7.3 są zaskakująco duże. W obrębie kuli indukowanej pięcioma scentrowanym sąsiadami znajduje się średnio ponad 53 konwencjonalnych sąsiadów, natomiast kula sąsiedztwa 10-NCN zawiera przeciętnie aż ok. 140 sąsiadów wg odległości. Zjawisko to sugeruje, iż ważność aspektu bliskości jest prawdopodobnie przeceniana. Co więcej, jeśli średnio tak wiele próbek położonych jest bliżej niż najdalszy z *k* scentrowanych sąsiadów, to może istnieć wiele sposobów „poprawy" zbioru sąsiadów celem lepszej predykcji etykiety próbki testowej. Do szukania innych reguł decyzyjnych, które brałyby pod uwagę oba opisy-

[13] W praktyce dość częste mogą być jednak wyjątki, np. wg cytowanej pracy Skalaka, klasyfikator „drugi najbliższy sąsiad" jest znacząco lepszy od klasyfikatora „pierwszy najbliższy sąsiad" na zbiorze Iris.

wane aspekty sąsiedztwa: bliskość i symetrię, zachęca heurystyczny wszakże charakter reguły *k*-NCN.



Rys. 7.3. Średnia liczba najbliższych sąsiadów w kuli wyznaczonej przez najdalszy spośród *k* scentrowanych sąsiadów, *k*=3..10, na przykładzie partycji nr 1 zbioru Ferrites

Ponieważ kryterium bliskości środka ciężkości z *k*-NCN wydaje się dość dobre ze względu na jakość predykcji, postanowiono pozostawić je bez zmian, optymalizując natomiast zbiór sąsiadów zgodnie z kryteriami: bliskości i odległości testowej próbki do środka ciężkości układu sąsiadów. Proponowany klasyfikator znajduje najpierw *k* scentrowanych sąsiadów, a następnie w pętli próbuje wymieniać losowych sąsiadów na inne próbki, o ile są one położone bliżej oraz lepiej w sensie opisanego kryterium bliskości środka ciężkości.

Dokładny opis klasyfikacji przy użyciu proponowanej przez autora rozprawy reguły decyzyjnej, którą nazwano *k Near Surrounding Neighbors*, zamieszczono poniżej. Idea zamiany sąsiadów jest analogiczna do zastosowanej przez Skalaka (1994) do selekcji prototypów do zredukowanego zbioru odniesienia.

Znajdź *k* scentrowanych sąsiadów testowej próbki *q*. Oznacz tych sąsiadów przez n_1, ..., n_k. Oznacz środek ciężkości układu sąsiadów przez *c*.
Dla *counter*=1 ... LICZBA_ITERACJI
{
Wybierz losowego sąsiada n_i, $1 \le i \le k$
Wybierz losową próbkę *s*, taką że $d(q,s) \le d\left(q,n_{najdalszy}\right)$,
gdzie $n_{najdalszy}$ jest *k*-tym, co do odległości, sąsiadem *q*

Jeśli $d(q,s) \le d(q,n_i)$
{
Niech *próbne_c* = środek_ciężkości(n_1, ..., n_{i-1}, s, n_{i+1}, ..., n_k)
Jeśli $d(q, próbne\_c) < d(q,c)$
{
Niech n_i=s
Niech c = *próbne_c*
}
}
}
Zwróć zbiór sąsiadów n_1, ..., n_k.

W eksperymentach autora faza uczenia dla *k*-NSN była przeprowadzana przy pomocy reguły *k*-NCN, co przyspiesza ten proces w stosunku do użycia *k*-NSN jako estymatora jakości dla poszczególnych wartości *k*.

Reguła *k*-NSN w pewnym sensie odracza część sesji uczenia do fazy klasyfikacji; jednym z pierwszych klasyfikatorów tego typu był schemat z klasyfikacją przeprowadzaną przy użyciu sieci neuronowej uczonej *on-line* na zbiorze złożonym z *k* najbliższych sąsiadów testowej próbki (Bottou i Vapnik, 1992). Choć oczywistą wadą tego schematu jest bardzo mała szybkość działania (w oryginalnej wersji, o kilka rzędów wielkości niższa od szybkości klasyfikacji *k*-NN), to jednak w eksperymentach w cytowanej pracy dotyczących rozpoznawania pisma ręcznego osiągnął on najwyższą jakość spośród kilku testowanych klasyfikatorów.

Probabilistyczna natura klasyfikatora *k*-NSN umożliwia skonstruowanie sieci klasyfikatorów *k*-NSN z prostym głosowaniem. Klasyfikacja dokonywana przez każdy komponent sieci zaczyna się od znalezienia *k* sąsiadów NCN, ale otrzymani w dalszym etapie klasyfikacji sąsiedzi NSN mogą się różnić między komponentami. Niestety, poprawa jakości względem pojedynczego klasyfikatora *k*-NSN w przeprowadzonych testach była minimalna i nieznacząca statystycznie.

7.3. Klasyfikatory grafowe

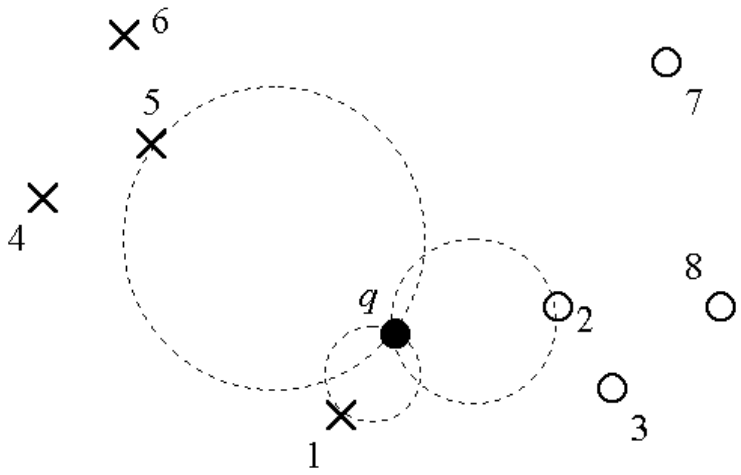
Definicje grafów i indukowanych nimi klasyfikatorów

Oprócz NCN rozważano w literaturze inne przykłady symetrycznego sąsiedztwa (Sánchez i in., 1997a; Zhang i in., 1997). Prawdopodobnie pierwsza definicja tego typu została podana przez O'Callaghana (1975), ale określona została ona tylko dla

przestrzeni R^2 (uogólnienie na wyższe wymiary nie jest łatwe) i była zależna od aż dwóch wolnych parametrów.

Przyjrzymy się bliżej koncepcji sąsiedztwa grafowego, a konkretniej sąsiedztwa indukowanego dwoma dobrze znanymi grafami bliskości: grafem Gabriela (ang. *Gabriel Graph*) i grafem pokrewieństwa (ang. *Relative Neighborhood Graph*). Oba grafy opisane są np. w pracy (Jaromczyk i Toussaint, 1992). Dla oznaczenia grafów będziemy używać skrótów: odpowiednio GG i RNG.

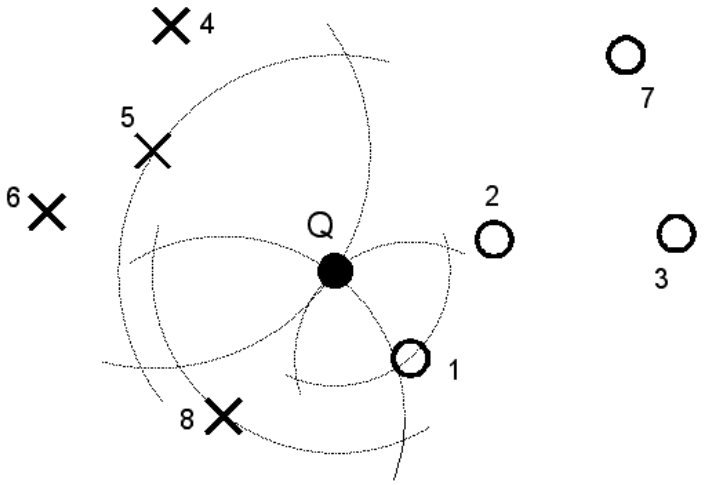
Mówiąc nieformalnie, dwa obiekty uważane są za sąsiadów Gabriela wtedy i tylko wtedy, gdy wnętrze kuli rozpiętej na tych obiektach nie zawiera żadnego obiektu ze zbioru odniesienia. Sąsiedzi Gabriela są w GG połączeni krawędzią. Rys. 7.4 przedstawia sąsiadów Gabriela próbki q, którymi są obiekty 1, 2 i 5.



Rys. 7.4. Sąsiedzi Gabriela próbki q

RNG jest podzbiorem zbioru GG. Kryterium jest bowiem silniejsze: dwa punkty x i y są pokrewnymi sąsiadami (ang. *relative neighbors*) wtedy i tylko wtedy, gdy żaden punkt nie leży bliżej zarówno do x jak i do y niż wynosi odległość od x do y. Na Rys. 7.5 pokrewnymi sąsiadami próbki Q są obiekty o indeksach 1, 5 i 8.

Oryginalnie oba grafy zostały zdefiniowane w metryce euklidesowej, nie widać jednak przeszkód dla rozważania także innych metryk. W rzeczy samej, do eksperymentów opisanych niżej autor używał zawsze metryki miejskiej.



Rys. 7.5. Pokrewni sąsiedzi próbki Q

Klasyfikatory indukowane przez GG i RGN (zwane także odpowiednio GGN i RNGN[14]) zostały zaproponowane w (Sánchez i in., 1997b). Ogólnie mówiąc, klasyfikator indukowany przez dany graf znajduje dla danej próbki wszystkich jej *m* sąsiadów grafowych ze zbioru odniesienia, a następnie przypisuje ją do klasy najliczniej reprezentowanej wśród tych *m* sąsiadów. Remisy rozstrzygane są arbitralnie (podobnie jak np. przy *k*-NN; popularnym rozwiązaniem w takiej sytuacji jest przypisanie próbki do najliczniejszej w całym zbiorze klasy spośród klas *ex aequo* zwycięskich). W testach przeprowadzonych przez Sáncheza i in. klasyfikatory GGN i RNGN zwykle pokonywały *k*-NN w sensie jakości, a nieraz nawet odnosiły zwycięstwo nad *k*-NCN. Podobnie jak w przypadku *k*-NCN, szybkość klasyfikacji może jednak być niestety zbyt mała w niektórych zastosowaniach.

Redukcja sąsiedztwa grafowego

W niniejszym punkcie zaproponowano modyfikację opisanych klasyfikatorów grafowych polegającą na odrzuceniu w czasie klasyfikacji części sąsiadów grafowych danej próbki. Idea autora rozprawy polega na połączeniu obu przedstawionych koncepcji symetrycznego sąsiedztwa: sąsiedztwa grafowego z regułą *k*-NCN, ale w zaproponowanej wersji lokalnej.

Konkretnie, dla danej próbki znajdowani są jej wszyscy sąsiedzi GG (RNG), a następnie wyłącznie w zbiorze znalezionych sąsiadów grafowych znajdowanych jest

[14] Końcowe „N” oznacza „Neighborhood” (sąsiedztwo).

k sąsiadów scentrowanych (tj. NCN) z *k* wybieranym lokalnie w taki sposób, aby odległość środka ciężkości rozpatrywanego układu była jak najbliższa testowej próbce. Otrzymany podzbiór używany jest do prostego głosowania w celu otrzymania finalnej etykiety klasy. Można powiedzieć nieformalnie, iż ideą propozycji jest otrzymanie „jeszcze bardziej symetrycznego" sąsiedztwa. Warto zauważyć, że podobnie jak oryginalne klasyfikatory grafowe, proponowany schemat również nie wymaga kosztownego uczenia — liczba sąsiadów nie zależy od żadnego współczynnika, ani też nie szacuje się przed klasyfikacją wartości *k*.

Sánchez i in. (1997b) twierdzą, iż zbyt wielu sąsiadów grafowych (co zdarza się często zwłaszcza z grafem GG) może źle wpłynąć na klasyfikację. Propozycja autora rozprawy redukuje liczbę sąsiadów grafowych z uwzględnieniem kryterium środka ciężkości układu.

Warto zastanowić się nad pewnymi formami przybliżonego sąsiedztwa grafowego. Choć jakość klasyfikacji jest pierwszoplanowym kryterium, efektem ubocznym może być większa szybkość klasyfikacji w takich schematach. Ogólny wzorzec otrzymania takiego klasyfikatora może wyglądać następująco: 1) wybierz pewną liczbę obiektów, które „wyglądają na dobre kandydatury" na sąsiadów; 2) zredukuj ten zbiór np. przy użyciu reguły *k*-NCN z lokalnymi wartościami *k*; 3) przypisz próbkę testową do klasy najliczniej reprezentowanej w otrzymanym zbiorze.

Hybrydy klasyfikatorów grafowych GGN i RNGN z lokalną regułą *k*-NCN przedstawiono w pracy (Grabowski, 2001b).

Wyniki testów klasyfikatorów grafowych

Przeprowadzono empiryczne porównanie opisanych metod na kilku rzeczywistych zbiorach danych. Dane zostały znormalizowane, używano zawsze metryki miejskiej. Remisy rozstrzygano na korzyść klasy z mniejszym numerem.[15]

Tab. 7.1 przedstawia błędy klasyfikatorów grafowych na poszczególnych dziesięciu losowych partycjach zbioru Ferrites oraz średnie i odchylenia standardowe błędów. Dla obu klasyfikatorów bazowych zaproponowana idea ograniczania sąsiedztwa przyniosła zarówno zmniejszenie błędu, jak i jego wariancji.

[15] Remisy często rozstrzyga się na korzyść najliczniejszej klasy. Na zbiorze Ferrites nie miało to jednak znaczenia, gdyż klasy w tym zbiorze ułożone są od największej liczebnie do najmniejszej.

Tabela 7.1: Błędy (w %) klasyfikatorów grafowych na zbiorze Ferrites

partycja	RNGN	RNGN + lokalna *k*-NCN	GGN	GGN + lokalna *k*-NCN
1	10,26	10,06	10,77	9,73
2	11,68	10,99	12,01	9,64
3	10,86	10,57	11,61	9,99
4	10,77	10,53	12,10	9,99
5	10,77	10,30	11,79	9,68
6	11,28	10,55	11,73	9,66
7	10,70	10,64	12,57	9,86
8	11,30	10,88	11,86	9,77
9	10,42	10,28	11,08	9,35
10	11,39	11,39	12,35	10,66
średnia	**10,94**	**10,62**	**11,79**	**9,83**
odch. std.	0,45	0,39	0,54	0,35

Wyniki drugiego testu prezentuje Tab. 7.2. Są to średnie błędy rozważanych klasyfikatorów na pięciu zbiorach UCI. Dodatkowo skonfrontowano wyniki klasyfikatorów grafowych z wynikami reguły *k*-NCN (ostatnia kolumna; wyniki *k*-NCN zostaną powielone w Tab. 7.5a). Tym razem widoczna jest duża rozbieżność wyników, i tak np. zaproponowana idea przynosi wyraźną poprawę jakości dla klasyfikatora GGN na zbiorze Wine, natomiast znacznie pogarsza jakość na zbiorze Pima. Średnie błędy po wszystkich pięciu zbiorach zostały zmniejszone tylko nieznacznie.

Tabela 7.2: Średni błąd (w %) klasyfikatorów grafowych oraz reguły *k*-NCN na zbiorach UCI

zbiór	RNGN	RNGN + lokalna *k*-NCN	GGN	GGN + lokalna *k*-NCN	*k*-NCN
Bupa	38,26	38,09	33,62	35,88	30,38
Glass	29,82	29,72	30,56	27,57	29,92
Iris	6,53	5,20	6,93	5,07	4,40
Pima	29,89	30,62	22,91	28,30	24,48
Wine	4,96	3,50	6,42	2,71	3,16
średnia	**21,89**	**21,43**	**20,09**	**19,91**	**18,47**

Oto wnioski, jakie można wysnuć z analizy wyników eksperymentów. Klasyfikatory grafowe są bardzo niestabilne w porównaniu z innymi klasyfikatorami. Choć w niektórych przypadkach jakość oferowana przez zaproponowaną hybrydę GGN z lokalną regułą *k*-NCN była bardzo dobra (zbiory Glass i Wine), to jednak średnio wszystkie algorytmy grafowe przegrywają z *k*-NCN i *k*-NSN (p. Rozdz. 7.6). Tym

niemniej, klasyfikatory z tej rodziny mają pewne zalety, które nie pozwalają na odrzucenie całej koncepcji.

Przede wszystkim, klasyfikatory grafowe nie wymagają uczenia, gdyż przy klasyfikacji nie korzystają z żadnego wyuczonego (ani też zadanego *ad hoc*) parametru. Prostota i elegancja koncepcji powinna również ułatwić analizę zachowania się algorytmu, na co można też spojrzeć od drugiej strony: porównanie danego klasyfikatora grafowego z innym klasyfikatorem może dać cenną informację o charakterze konkretnego zadania.

Po drugie, wstępne eksperymenty (wyniki nie zamieszczone w rozprawie) sugerują, iż klasyfikatory grafowe są stosunkowo odporne na zbędne cechy — zjawisko to wymaga jednak dokładniejszego zbadania.

7.4. Schemat z głosowaniem dla różnych wartości parametru *k*

Zaproponowany w niniejszym podrozdziale algorytm jest dość ogólnym schematem klasyfikacji z głosowaniem klasyfikatorów tego samego typu używających do klasyfikacji pewnej liczby sąsiadów *k*. Ponieważ w pracy opisano trzy proste reguły decyzyjne działające w oparciu o głosowanie *k* sąsiadów (*k* – niezmienne w obrębie danego zadania), zatem z przedstawionego poniżej wzorca można wyprowadzić trzy konkretne algorytmy, które nazwano *voting k*-NN (Grabowski, 2002b), *voting k*-NCN (Grabowski, Jóźwik i Chen, 2003; Grabowski i Sokołowska, 2003) i *voting k*-NSN (Grabowski, Jóźwik i Chen, 2003).

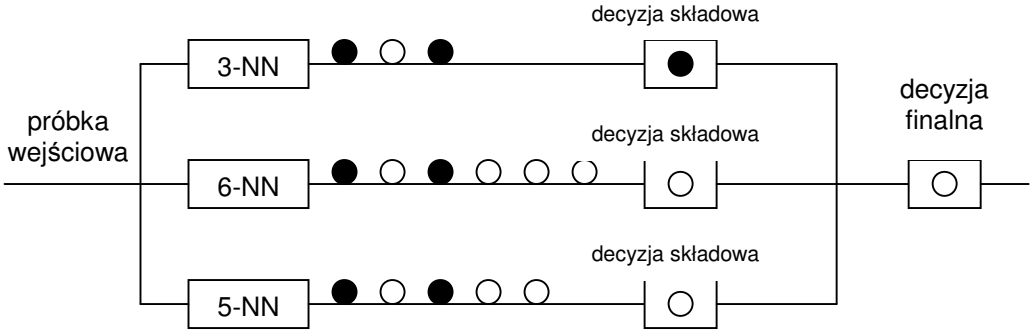
Klasyfikator *voting k*-NN

Wadą reguły *k*-NN w przypadku skończonym jest konieczność wyboru odpowiedniej wartości *k*, co jest zwykle realizowane przy pomocy metody minus jednego elementu i nie gwarantuje optymalnej jakości klasyfikacji na zbiorze testowym. Innymi słowy, pomimo dużej prostoty nawet reguła *k*-NN nie jest zabezpieczona przed przeuczeniem, gdyż estymowana optymalna wartość parametru *k* nie musi naprawdę implikować najlepszej jakości w zadaniu.

Klasyfikator *voting k*-NN próbuje przezwyciężyć opisaną wadę *k*-NN poprzez użycie zespołu wielu klasyfikatorów *k*-NN działających z tym samym zbiorem odniesienia, ale z wyselekcjonowanymi osobno wartościami *k*. Predykcje poszczególnych klasyfikatorów *k*-NN są uśredniane w wyniku prostego głosowania. Komponenty zespołu

uczone są na losowych partycjach całego zbioru uczącego. Nietrudno zauważyć, że pewną wadą schematu w stosunku do zwykłej *k*-NN jest dłuższy czas uczenia (zjawisko niemal zawsze występujące przy zespołach klasyfikatorów).

W Rozdz. 7.2 podano, iż reguła *k*-NN może być postrzegana jako klasyfikator równoległy. W pewnym ujęciu zatem, *voting k*-NN jest klasyfikatorem hierarchicznym, tj. z głosowaniem w dwóch etapach. Przykład działania schematu *voting k*-NN dla sieci trzech komponentów ukazany jest na Rys. 7.6. Wejściowa próbka zostanie przypisana do klasy „białej", bo tak wskazują dwa z trzech klasyfikatorów składowych.



Rys. 7.6. Podejmowanie decyzji w schemacie *voting k*-NN na przykładzie zespołu trzech klasyfikatorów z wyuczonymi wartościami *k* równymi 3, 6 i 5

Powstaje pytanie, jak skonstruować (nauczyć) składowe klasyfikatory *k*-NN, aby zapewnić zarazem pewną ich różnorodność, jak i stosunkowo wysoką jakość klasyfikacji. Znaną techniką pozwalającą ocenić jakość klasyfikatora w czasie uczenia (czyli na etapie projektowania) jest podział całego zbioru uczącego na część tzw. Konstrukcyjną (ang. *construction set*) oraz część walidacyjną (ang. *validation set*). Rozważany na etapie projektowania klasyfikator jest uczony na zbiorze konstrukcyjnym, zaś testowany na zbiorze walidacyjnym. Oczywistą wadą tej techniki jest zmniejszenie się właściwego zbioru uczącego. Wydaje się jednak, iż jest to cena, którą trzeba zapłacić za komfort pewnej oceny klasyfikatora już w fazie uczenia. Powstaje jednak problem: jak podzielić dany zbiór na opisane dwie części? Zbyt mało próbek w zbiorze konstrukcyjnym powoduje bardzo słabą aproksymację rzeczywistego rozkładu prawdopodobieństw w zadaniu. Z drugiej strony, zbyt niewielki zbiór walidacyjny oznacza niepewną estymację wygenerowanego klasyfikatora.

Autor rozprawy podjął decyzję, aby podzielić zbiór uczący (odniesienia) na równe części (połowy). Szukano następnie optymalnej wartości *k* testując wyniki klasyfikacji próbek z części walidacyjnej w odniesieniu do części konstrukcyjnej. Opisany podział

zbioru odniesienia przeprowadzony został losowo L razy. W wyniku uczenia komponentów otrzymano wartości k_1, ..., k_L.

Procedura uczenia zaprezentowana jest także poniżej.

```
// Tr – zbiór uczący (treningowy)
Dla i=1 ... L     // L prób
{
      Niech RS(i) = losowy_wybor(rozmiar(Tr) / 2);
      // RS(i) – losowy podzbiór Tr
      CVS(i) = Tr \ RS(i);     // CVS(i) – aktualny zbiór walidacyjny
      k(i) = znajdz_optymalne_k(RS(i), CVS(i));
      // na zbiorze RS(i) z odniesieniem do zbioru walidacyjnego CVS(i)
}
// wyjście: wyuczone parametry k(1), ..., k(L)
```

Klasyfikacja próbki q polega na znalezieniu L etykiet klas dla q zgodnie z regułami k_i-NN, i=1..L, a następnie przypisaniu q do klasy najliczniej występującej wśród znalezionych etykiet.

Jak wspomniano wcześniej, wprowadzono także klasyfikatory *voting* k-NCN i *voting* k-NSN, bazujące odpowiednio na regułach k-NCN oraz k-NSN, i będące pełną analogią do schematu *voting* k-NN.

Złożoność obliczeniowa klasyfikacji

Przy założeniu, że nie korzysta się ze specjalnych struktur danych (które, jak wiadomo, i tak są mało efektywne w wysokich wymiarach), możliwe są dwie implementacje procedury znajdowania k najbliższych sąsiadów. W praktyce zazwyczaj używa się implementacji *naiwnej*. Podczas szukania trzymana jest lista k najbliższych sąsiadów danej próbki, która jest w razie potrzeby uaktualniana. Chociaż typowo (i dla małych wartości k) czas szukania jest rzędu 1,1dn ... 1,2dn, gdzie d jest wymiarem (tj. zaledwie ok. 10%–20% dłuższy niż czas szukania jednego najbliższego sąsiada), to w najgorszym przypadku będzie on rzędu $O(dnk)$. Aby zabezpieczyć się przed najgorszym przypadkiem, można użyć innej implementacji. W pierwszym kroku dla danej próbki liczone są odległości do wszystkich próbek ze zbioru odniesienia, a następnie są one sortowane.[16] Głosowanie po k pierwszych sąsiadach daje odpowiedź klasyfikatora.

[16] Nie jest nawet konieczne pełne sortowanie.

Czas klasyfikacji wynosi teraz zatem $O(dn + n\log n + k + c)$, c – liczba klas. Ponieważ $k \le n$ i $c \le n$ (zazwyczaj $k << n$ i $c << n$), zaś log n jest zwykle rzędu wielkości d, to ogólny koszt jest zbliżony do $O(dn)$.

Klasyfikacja w schemacie *voting* k-NN może wykorzystywać pierwszą lub drugą ideę implementacyjną. W obu przypadkach zasadnicze *novum* polega na braniu pod uwagę $k_{\max} := \max_{i=1..L} k_i$ sąsiadów. Koszt klasyfikacji przy drugiej z opisanych, bezpiecznej, implementacji wynosi zatem $O(dn + n\log n + k_{\max} + Lc)$. Dla umiarkowanych wartości L (w przeprowadzonych testach przyjęcie L=10 prowadziło do dość dobrych wyników; p. Rozdz. 7.6) koszt jest praktycznie porównywalny z kosztem klasyfikacji oryginalnej reguły k-NN w tej samej wersji implementacyjnej.

W przypadku klasyfikatora *voting* k-NCN, spowolnienie względem reguły bazowej wyraża się stosunkiem wartości $k_{\max}$ do k wybranego globalnie, gdzie $k_{\max}$ jest największą liczbą sąsiadów spośród wartości wybranych dla L komponentów. Czynnik spowolnienia dla reguły *voting* k-NSN jest bliski podanemu ilorazowi dla *voting* k-NCN.

7.5. Klasyfikatory kaskadowe wykorzystujące koncepcję symetrycznego sąsiedztwa

Przypomnijmy, że klasyfikatory kaskadowe (Rozdz. 3.2) przeprowadzają klasyfikację w kilku etapach, przy czym nie wszystkie próbki muszą „dojść" aż do ostatniego etapu. Pierwszy etap jest najprostszym i najszybszym klasyfikatorem — jeśli kryterium oceny próbki uzna, iż jest ona wystarczająco „łatwa", to klasyfikacja danej próbki jest zakończona. W przeciwnym razie próbka przechodzi do następnego etapu itd. — najtrudniejsze próbki zostają zaklasyfikowane w ostatnim etapie.

Ponieważ reguły decyzyjne k-NCN i k-NSN, a także ich wersje z głosowaniem po różnych wartościach k (*voting* k-NCN i *voting* k-NSN), oferują wysoką jakość klasyfikacji (płacąc za to małą szybkością), więc zasadne jest ich użycie w dalszych etapach klasyfikatora kaskadowego. Zaproponowano niżej kilka schematów dwuetapowego klasyfikatora kaskadowego korzystającego z reguł k-NCN lub k-NSN.

Schemat z siecią zbiorów zredukowanych i regułą *k*-NCN (Grabowski, 2003a)

Niewątpliwą wadą reguły *k*-NCN jest mała prędkość klasyfikacji. Złożoność klasyfikacji pojedynczej próbki wynosi $O(kdn)$, co negatywnie kontrastuje ze złożonością bliską $O(dn)$ dla *k*-NN w typowym przypadku. Kolejnym celem autora było skonstruowanie klasyfikatora szybszego niż *k*-NCN, ale oferującego porównywalną (lub wyższą) jakość predykcji.

Ponieważ zdecydowano, iż próbki „trudne" będą klasyfikowane przy użyciu efektywnej reguły *k* scentrowanych sąsiadów, więc klasyfikacja w etapie pierwszym musi być znacząco szybsza. Postanowiono użyć klasyfikatora bazującego na regule jednego najbliższego sąsiada (1-NN) i wykorzystującego redukcję zbioru odniesienia. Interesującą koncepcją poprawy jakości w tego typu metodach jest głosowanie większościowe wielu (przynajmniej 3) zredukowanych zbiorów odniesienia (Skalak, 1997). Co więcej, użycie kilku zbiorów zredukowanych posłuży do wskazania prostego kryterium oceny wiarygodności predykcji w pierwszej fazie klasyfikatora. Konkretnie: jeśli etykieta klasy l zwracana przez wszystkie L klasyfikatorów D_1, ..., D_L typu 1-NN działających ze zredukowanym zbiorem odniesienia będzie taka sama, to próbka zostanie uznana za „łatwą" i zostanie przypisana do klasy sugerowanej przez klasyfikatory D_i, $i = 1..L$. Jeśli zaś nie, to próbkę odeśle się do klasyfikatora drugiego etapu, tj. *k*-NCN. Dokładniej algorytm uczenia i klasyfikacji proponowanego schematu kaskadowego z wykorzystaniem reguł 1-NN i *k*-NCN opisany jest poniżej.

Uczenie (*L*)

1. Wygeneruj niezależnie od siebie L klasyfikatorów D_1, ..., D_L przy użyciu zbioru odniesienia S zawierającego obiekty z przestrzeni X. Dodatkowo, niech D_1, ..., D_L korzystają ze zredukowanego zbioru odniesienia i reguły 1-NN.
2. Wyznacz optymalne k dla reguły *k*-NCN przy użyciu zbioru odniesienia S.

Klasyfikacja próbki *q*:

1. Sklasyfikuj q klasyfikatorami D_1, ..., D_L, otrzymując etykiety klas $l_q(i)$, i=1..L.
2. Jeśli $l_q(1) = l_q(2) = ... = l_q(L)$, to przypisz q do klasy $l_q(1)$. W przeciwnym razie przekaż q na wejście klasyfikatora *k*-NCN z wyuczoną wcześniej wartością k i przypisz q do klasy wskazanej przez ten klasyfikator.

Należy zauważyć, że jeśli błędy popełniane przez klasyfikatory D_1, ..., D_L są niezależne i każdy klasyfikator ma takie samo prawdopodobieństwo pomyłki p, to

prawdopodobieństwo iż L jednakowych etykiet zwracanych przez $D_1, ..., D_L$ oznacza błędną predykcję jest nie większe niż p^L (dokładnie równe tej wartości w przypadku dwóch klas i być może mniejsze, jeśli liczba klas jest większa niż 2), co przy odpowiednio małej wartości p dąży szybko do zera wraz ze zwiększaniem liczby klasyfikatorów składowych. Niestety — niezależność klasyfikatorów składowych w praktyce nie jest osiągana, a wytłumaczyć ten fenomen można w kolokwialny sposób: „trudne" próbki są na ogół trudne dla wszystkich klasyfikatorów składowych. Niemniej jednak, intuicja podpowiada, że prawdopodobieństwo błędnej decyzji wszystkich klasyfikatorów w zespole musi być małe, jeśli klasyfikatory te były uczone osobno, zaś sesje uczenia były niezależne i nie zawierały przeglądu zbyt wielu modeli kandydujących.

Do generacji klasyfikatorów $D_1, ..., D_L$ zastosowano jeden z algorytmów redukcji Skalaka (1994), opisany dokładniej w Rozdz. 6.2 i oznaczony skrótem RMHC (*Random Mutation Hill Climbing*). Należy przypomnieć, że algorytm zależny jest od dwóch wybranych przez użytkownika parametrów: liczności zbioru zredukowanego h i liczby mutacji m. Parametr m mieści się zazwyczaj w przedziale 100–1000 (zbyt duża wartość może prowadzić do przeuczenia).

Inne klasyfikatory kaskadowe (Grabowski, 2003a)

Szereg następnych zaproponowanych klasyfikatorów można uznać za jedną rodzinę ze względu na identyczny schemat ich konstrukcji. W pierwszym etapie występuje zespół klasyfikatorów, a ich decyzja uważana jest za ostateczną wtedy i tylko wtedy, gdy wszystkie głosy są identyczne. W sytuacji braku jednomyślności używany jest (wolniejszy) klasyfikator drugiego etapu. Ów drugi klasyfikator również może mieć postać równoległą, ale tym razem jednomyślność głosów nie ma już znaczenia.

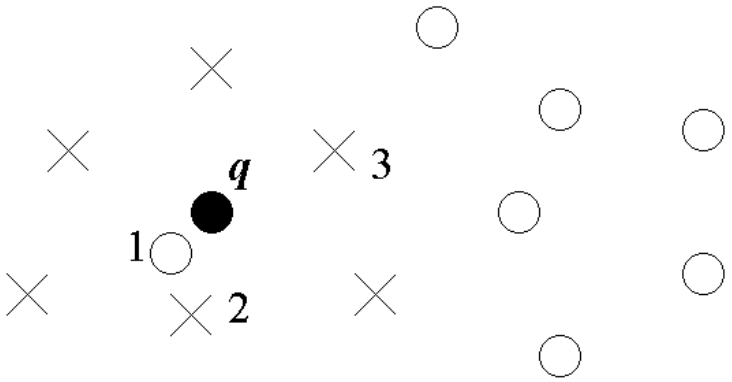
Proponowane klasyfikatory to:

- *voting k*-NN + *k*-NCN;
- *voting k*-NN + *k*-NSN;
- *voting k*-NN + *voting k*-NCN;
- *voting k*-NN + *voting k*-NSN.

Celem autora rozprawy nie było uzyskanie wyższej jakości klasyfikacji niż ta, którą oferuje klasyfikator z drugiego etapu traktowany pojedynczo. Eksperyment zostaje uznany za pomyślny, jeśli wyniki klasyfikatorów kaskadowych będą porówny-

walne z wynikami klasyfikatorów z drugiego etapu, przy założeniu, że do drugiego etapu nie wchodzi znaczący odsetek testowych próbek (np. 50% lub więcej). Ponieważ pierwszy etap jest zazwyczaj przynajmniej kilkakrotnie szybszy niż drugi, więc w opisanej sytuacji otrzymywane jest znaczące przyspieszenie klasyfikacji w stosunku do osobnego klasyfikatora z drugiego etapu.

We wszystkich wyliczonych powyżej schematach kaskadowych pierwszy etap stanowi klasyfikator *k*-NN z głosowaniem po różnych wartościach *k*. Zespół takich klasyfikatorów może uzyskać consensus głosów nawet przy pewnym poziomie szumu w otoczeniu testowanej próbki. Zjawisko to ilustruje Rys. 7.7. Mimo iż najbliższym sąsiadem próbki testowej *q* jest obiekt nr 1 z klasy „kółek" (ewidentnie będący szumem), to jednak istnieje szansa, że odpowiedź wszystkich klasyfikatorów składowych pierwszego etapu klasyfikatora kaskadowego, tj. *voting k*-NN, będzie identyczna i klasyfikacja zakończy się w pierwszym etapie. Dokładniej, próbka *q* może być poprawnie sklasyfikowana przez wszystkie *L* komponenty typu k_i-NN pierwszego etapu (tj. przypisana do klasy „krzyżyków"), o ile $k_i \geq 3$, $i = 1..L$.



Rys. 7.7. Demonstracja odporności na szum zespołu klasyfikatorów typu *voting k*-NN

Gdyby pierwszym etapem klasyfikatora kaskadowego była reguła *k*-NN pojmowana jako klasyfikator równoległy, to znacznie więcej próbek musiałoby przechodzić do drugiego, bardziej kosztownego, etapu.

7.6. Wyniki eksperymentów i dyskusja

Autor niniejszej rozprawy przeprowadził szereg testów opisanych metod na następujących zbiorach danych: Ferrites, Remotes, Bupa, Glass, Iris, Pima i Wine. Porównano następujące algorytmy:

- *k*-NN;
- *voting k*-NN;
- *k*-NCN;
- *voting k*-NCN;
- *k*-NSN (przy różnej liczbie mutacji);
- *voting k*-NSN (przy różnej liczbie mutacji);
- sieć równoległa zbiorów zredukowanych Skalaka + *k*-NCN;
- *voting k*-NN + *k*-NCN;
- *voting k*-NN + *k*-NSN;
- *voting k*-NN + *voting k*-NCN;
- *voting k*-NN + *voting k*-NSN.

Powyższe algorytmy, z wyjątkiem oryginalnych reguł *k*-NN i *k*-NCN, zostały zaproponowane przez autora rozprawy. Ponadto na zbiorach Ferrites i Remotes przetestowano algorytmy *k*-NCN i *k*-NSN w dwóch wariantach dekompozycyjnych: Jóźwik–Vernazza oraz Moreira–Mayoraz (p. Rozdz. 3.3). Każda para klas uczestniczących w głosowaniu używała w tych algorytmach własnej wyuczonej wartości parametru *k*.

We wszystkich eksperymentach dane zostały poddane wstępnej standaryzacji (klasycznej, tj. z użyciem odchylenia standardowego), używano zawsze metryki miejskiej. Remisy rozstrzygano na korzyść klasy z mniejszym numerem.

Tab. 7.3 zawiera wyniki (średnie błędy i odchylenia standardowe z 10 losowych partycji) algorytmów na zbiorze Ferrites. Litery PW w odpowiednich wierszach Tabel 7.3 i 7.4 oznaczają schematy dekompozycyjne (od ang. *PairWise*).

Tabela 7.3: Średnie i odchylenia standardowe błędów testowanych klasyfikatorów na zbiorze Ferrites

klasyfikator	średnia [%]	odch. std. [%]
k-NN	10,96	0,77
k-NCN	10,18	0,82
k-NSN, 500 mut.	9,98	0,85
k-NSN, 2500 mut.	9,94	0,86
voting *k*-NN	10,42	0,56
voting *k*-NCN	9,96	0,80
voting *k*-NSN, 500 mut.	9,38	0,87
voting *k*-NSN, 2500 mut.	9,45	0,84
voting *k*-NN + *k*-NCN	9,99	0,88
voting *k*-NN + *k*-NSN, 500 mut.	9,88	0,92
voting *k*-NN + voting *k*-NCN	9,64	0,80
voting *k*-NN + voting *k*-NSN, 500 mut.	9,45	0,84
PW (J–V) *k*-NN	10,51	0,76
PW (M–M) *k*-NN	10,24	0,61
PW (J–V) *k*-NCN	9,66	0,59
PW (M–M) *k*-NCN	9,76	0,45
PW (J–V) *k*-NSN, 500 mut.	9,58	0,42
PW (M–M) *k*-NSN, 500 mut.	9,42	0,43
PW (J–V) *k*-NSN, 2500 mut.	9,00	0,28
PW (M–M) *k*-NSN, 2500 mut.	9,12	0,25

Tab. 7.4 przedstawia wyniki dla zbioru Remotes, przy trzech licznościach zbiorów uczących: po 100, 150 i 250 próbek na klasę. Kolejne wiersze zawierają średnie błędy i odchylenia standardowe dla testowanych klasyfikatorów. Ostatnia kolumna jest szacunkowym (rzecz jasna, względnym) określeniem typowej szybkości klasyfikacji danego algorytmu. Określenie szybkości typu „średnia/mała” dla klasyfikatorów kaskadowych oznacza, że szybkość ta zależy od zbioru: jeśli etykiety zdecydowanej większość próbek zostaną określone w pierwszym etapie klasyfikacji, to szybkość będzie „średnia”; w przeciwnym razie szybkość ta może być „mała”.

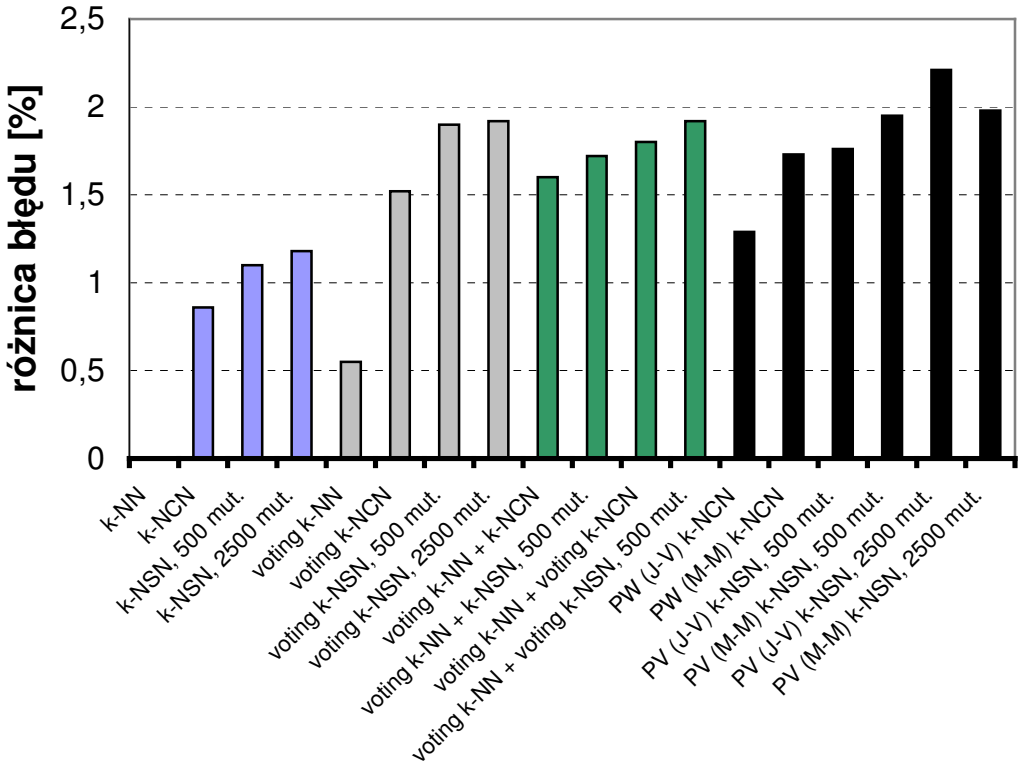
Dodatkowo, na Rys. 7.8 przedstawiono w postaci graficznej różnice średnich błędów między klasyfikatorem *k*-NN (pojmowanym jako wzorzec odniesienia) a pozostałymi testowanymi klasyfikatorami na zbiorze Remotes250. Słupki odpowiadające klasyfikatorom różnych typów („pojedyncze”, typu *voting*, kaskadowe i z dekompozycją na pary klas) zostały oznaczone różnymi odcieniami szarości.

Tabela 7.4: Średnie i odchylenia standardowe błędów testowanych klasyfikatorów na zbiorze Remotes

liczba próbek na klasę →	100		150		250		szybkość klasyfikacji
klasyfikator	błąd [%]		błąd [%]		błąd [%]		
k-NN	śred.	**24,81**	śred.	**23,32**	śred.	**21,63**	średnia
	odch. std.	0,65	odch. std.	0,72	odch. std.	0,66	
k-NCN	śred.	**24,30**	śred.	**21,97**	śred.	**20,77**	mała
	odch. std.	1,18	odch. std.	0,59	odch. std.	0,69	
k-NSN, 500 mut.	śred.	**23,86**	śred.	**21,61**	śred.	**20,53**	mała
	odch. std.	0,65	odch. std.	0,53	odch. std.	0,83	
k-NSN, 2500 mut.	śred.	**23,78**	śred.	**21,81**	śred.	**20,45**	mała
	odch. std.	0,76	odch. std.	0,69	odch. std.	0,88	
voting *k*-NN	śred.	**24,07**	śred.	**22,62**	śred.	**21,08**	średnia
	odch. std.	0,42	odch. std.	0,74	odch. std.	0,66	
voting *k*-NCN	śred.	**23,12**	śred.	**21,32**	śred.	**20,11**	mała
	odch. std.	0,74	odch. std.	0,61	odch. std.	0,56	
voting *k*-NSN, 500 mut.	śred.	**22,43**	śred.	**20,95**	śred.	**19,73**	mała
	odch. std.	0,57	odch. std.	0,59	odch. std.	0,62	
voting *k*-NSN, 2500 mut.	śred.	**22,62**	śred.	**21,08**	śred.	**19,71**	mała
	odch. std.	0,78	odch. std.	0,59	odch. std.	0,59	
voting *k*-NN + *k*-NCN	śred.	**23,13**	śred.	**21,44**	śred.	**20,03**	średnia/mała
	odch. std.	0,93	odch. std.	0,47	odch. std.	0,64	
voting *k*-NN + *k*-NSN, 500 mut.	śred.	**22,88**	śred.	**21,33**	śred.	**19,91**	średnia/mała
	odch. std.	0,76	odch. std.	0,47	odch. std.	0,67	
voting *k*-NN + voting *k*-NCN	śred.	**22,64**	śred.	**21,25**	śred.	**19,83**	średnia/mała
	odch. std.	0,69	odch. std.	0,58	odch. std.	0,60	
voting *k*-NN + voting *k*-NSN, 500	śred.	**22,37**	śred.	**21,02**	śred.	**19,71**	średnia/mała
	odch. std.	0,53	odch. std.	0,57	odch. std.	0,55	
PW (J–V) *k*-NCN	śred.	**23,51**	śred.	**21,71**	śred.	**20,34**	bardzo mała
	odch. std.	0,60	odch. std.	0,92	odch. std.	0,79	
PW (M–M) *k*-NCN	śred.	**22,98**	śred.	**21,13**	śred.	**19,90**	bardzo mała
	odch. std.	0,66	odch. std.	0,66	odch. std.	0,58	
PW (J–V) *k*-NSN, 500 mut.	śred.	**22,70**	śred.	**21,15**	śred.	**19,87**	bardzo mała
	odch. std.	0,82	odch. std.	0,60	odch. std.	0,66	
PW (M–M) *k*-NSN, 500 mut.	śred.	**22,55**	śred.	**20,89**	śred.	**19,68**	bardzo mała
	odch. std.	0,81	odch. std.	0,80	odch. std.	0,53	
PW (J–V) *k*-NSN, 2500 mut.	śred.	**22,42**	śred.	**20,74**	śred.	**19,42**	bardzo mała
	odch. std.	0,87	odch. std.	0,70	odch. std.	0,59	
PW (M–M) *k*-NSN, 2500 mut.	śred.	**22,67**	śred.	**20,89**	śred.	**19,65**	bardzo mała
	odch. std.	0,70	odch. std.	0,62	odch. std.	0,54	

Tab. 7.5a, b i c przedstawiają średnie błędy algorytmów na pięciu zbiorach UCI. Metodologia testów w tym przypadku była nieco inna. Pięć razy przeprowadzono 5-krotną walidację skrośną (ang. *5-fold cross validation*), a zatem błędy w Tabelach są średnimi z 25 otrzymanych dla każdej pary (zbiór, klasyfikator) wyników. Dodatkowo przetestowano wspomnianą w Rozdz. 7.2 wersję *k*-NSN z głosowaniem po pięciu znalezionych różnych sąsiedztwach NSN (z tą samą jednak wartością *k*), oznaczaną w tabelach przez v*k*-NSN. Dla zbiorów UCI nie przetestowano wersji dekompozycyjnych J–V i M–M, gdyż największe z tych zbiorów, tj. Pima i Bupa, przedstawiają

zadania dwudecyzyjne, zaś pozostałe zbiory są dość małe (ok. 200 obiektów lub mniej), a zatem niebezpieczeństwo przeuczenia jest znaczne. Co więcej, 216-elementowy zbiór Glass złożony jest aż z sześciu klas.



Rys. 7.8. Bezwzględne różnice średniego błędu między klasyfikatorem *k*-NN a pozostałymi testowanymi klasyfikatorami na zbiorze Remotes250. Większe wartości oznaczają wyższą jakość danego klasyfikatora

Tabela 7.5a: Średnie błędy (w %) klasyfikatorów „pojedynczych" na zbiorach UCI

	1	2	3	4	5
zbiór	***k*-NN**	***k*-NCN**	***k*-NSN, 100**	***k*-NSN, 500**	***k*-NSN, 2500**
Bupa	35,54	30,38	30,90	31,48	33,16
Glass	28,80	29,92	28,62	28,06	28,06
Iris	5,87	4,40	4,67	4,93	4,93
Pima	24,73	24,48	24,22	24,95	25,42
Wine	3,04	3,16	2,82	2,26	2,37
średnia	**19,60**	**18,47**	**18,25**	**18,34**	**18,79**

Tabela 7.5b: Średnie błędy (w %) klasyfikatorów z głosowaniem po różnych wartościach *k* na zbiorach UCI

	6	7	8	9	10
zbiór	**voting *k*-NN**	**voting *k*-NCN**	**voting *k*-NSN, 100**	**voting *k*-NSN, 500**	**voting *k*-NSN, 2500**
Bupa	34,20	30,43	30,49	30,78	32,75
Glass	28,43	28,69	25,99	26,18	26,18
Iris	5,47	4,00	4,13	3,87	4,67
Pima	24,71	24,16	24,53	24,16	24,37
Wine	3,37	2,36	2,03	2,36	2,37
średnia	**19,24**	**17,93**	**17,43**	**17,47**	**18,07**

Duża ilość zgromadzonych wyników utrudnia ich analizę. Aby ułatwić tę czynność, w oparciu o dane z Tabel 7.5a–c policzono dla poszczególnych algorytmów sumy rang na poszczególnych zbiorach. Algorytm o najmniejszym średnim błędzie dla danego zbioru otrzymał rangę 0, drugi pod względem jakości algorytm — rangę 1 itd. W przypadku remisów, rangi były równe, np. jeśli na danym zbiorze pewne dwa algorytmy osiągnęły najwyższą jakość, to otrzymywały one rangę 0,5. W Tab. 7.6 podane są rangi poszczególnych metod na pojedynczych zbiorach UCI (kolumny 2–6) oraz ich sumy (kolumna 7). Dodatkowo w kolumnie 8 podano ogólną rangę metody, zaczynając numerację od 1.

Tabela 7.5c: Średnie błędy (w %) klasyfikatorów kaskadowych oraz wersji *k*-NSN z głosowaniem na zbiorach UCI

	11	12	13	14	15	16	17	18
zbiór	**5·Skalak + *k*-NCN**	**voting *k*-NN + *k*-NCN**	**voting *k*-NN + voting *k*-NCN**	**voting *k*-NN + *k*-NSN, 500**	**voting *k*-NN + voting *k*-NSN, 500**	**v*k*-NSN, 100**	**v*k*-NSN, 500**	**v*k*-NSN, 2500**
Bupa	31,19	30,84	30,78	30,67	30,96	31,42	30,67	32,99
Glass	28,04	28,60	28,04	27,50	26,46	28,25	28,43	28,25
Iris	4,53	4,53	4,27	4,80	4,53	4,53	4,67	4,93
Pima	24,11	23,90	24,14	23,98	24,11	24,43	24,58	24,61
Wine	3,03	2,26	1,92	1,92	2,14	3,04	2,48	2,03
średnia	**18,18**	**18,03**	**17,83**	**17,77**	**17,64**	**18,33**	**18,17**	**18,56**

Tabela 7.6: Rangi testowanych algorytmów: zbiorcze i na poszczególnych zbiorach danych

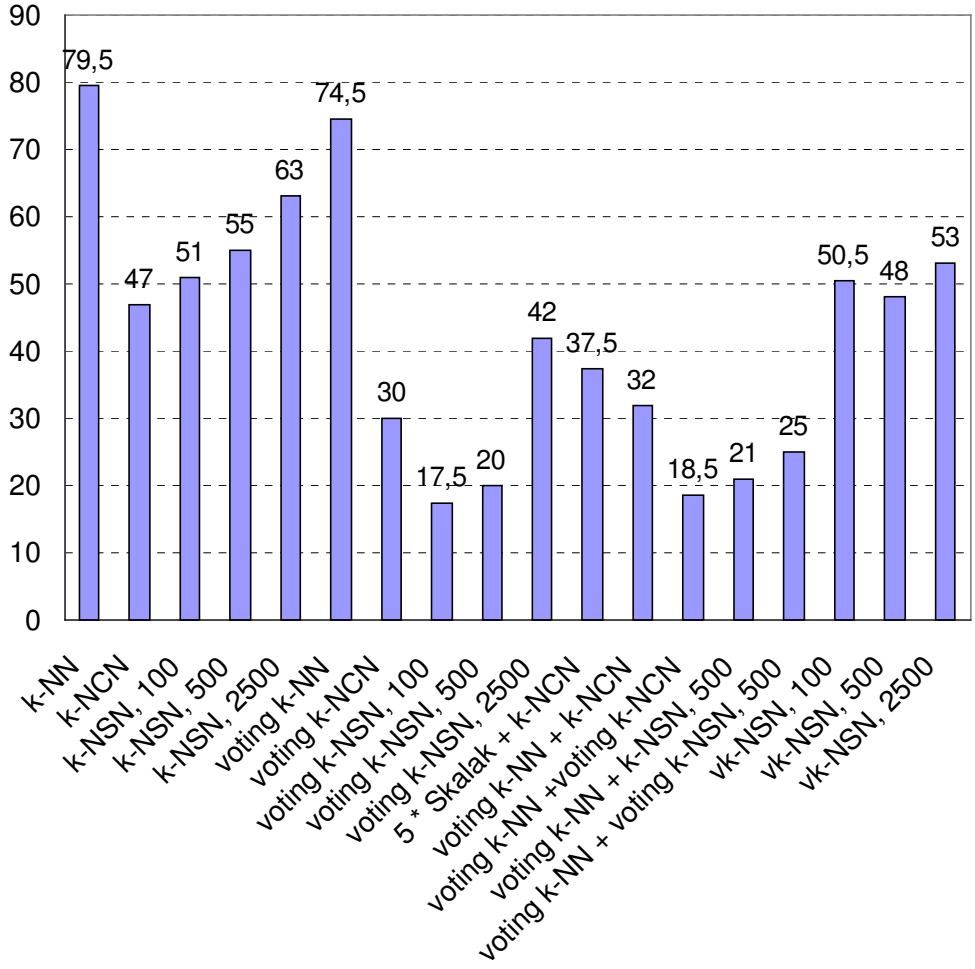
1	2	3	4	5	6	7	8
zbiór danych ► algorytm ▼	**Bupa**	**Glass**	**Iris**	**Pima**	**Wine**	**suma rang**	**ranga metody**
k-NN	17	16	17	15	14,5	**79,5**	**18**
k-NCN	0	17	4	10	16	**47**	**10**
k-NSN, 100	8	14	10	7	12	**51**	**13**
k-NSN, 500	12	7,5	14	16	5,5	**55**	**15**
k-NSN, 2500	15	7,5	14	17	9,5	**63**	**16**
voting *k*-NN	16	11,5	16	14	17	**74,5**	**17**
voting *k*-NCN	1	15	1	5,5	7,5	**30**	**6**
voting *k*-NSN, 100	2	0	2	11	2,5	**17,5**	**1**
voting *k*-NSN, 500	5,5	1,5	0	5,5	7,5	**20**	**3**
voting *k*-NSN, 2500	13	1,5	10	8	9,5	**42**	**9**
5 · Skalak + *k*-NCN	10	5,5	6,5	2,5	13	**37,5**	**8**
voting *k*-NN + *k*-NCN	7	13	6,5	0	5,5	**32**	**7**
voting *k*-NN + voting *k*-NCN	5,5	5,5	3	4	0,5	**18,5**	**2**
voting *k*-NN + *k*-NSN, 500	3,5	4	12	1	0,5	**21**	**4**
voting *k*-NN + voting *k*-NSN, 500	9	3	6,5	2,5	4	**25**	**5**
v*k*-NSN, 100	11	9,5	6,5	9	14,5	**50,5**	**12**
v*k*-NSN, 500	3,5	11,5	10	12	11	**48**	**11**
v*k*-NSN, 2500	14	9,5	14	13	2,5	**53**	**14**

Sumy rang przetestowanych algorytmów są dodatkowo zaprezentowane w formie graficznej na Rys. 7.9.

Drugim podstawowym kryterium oceny klasyfikatora, obok jakości klasyfikacji, jest jej szybkość. W przypadku klasyfikatorów hybrydowych pewną miarą szybkości jest odsetek próbek rozstrzyganych w każdym etapie. Dodatkowo, warto zwrócić uwagę na błędy predykcji w pierwszym oraz drugim etapie. Tab. 7.7a i 7.7b przedstawiają statystyki dotyczące klasyfikatorów „5 · Skalak + *k*-NCN" oraz „*voting k*-NN + *voting k*-NSN", tj. odpowiednio średnio najgorszego i najlepszego, w sensie jakości predykcji, spośród zaproponowanych klasyfikatorów kaskadowych.

Tabela 7.7a: Statystyki dotyczące klasyfikatora kaskadowego „5·Skalak + *k*-NCN"

zbiór	**błąd ogółem [%]**	**% próbek w II etapie**	**błąd w I etapie [%]**	**błąd w II etapie [%]**
Bupa	31,19	67,07	27,64	32,93
Glass	28,04	47,48	11,03	46,85
Iris	4,53	9,33	1,47	34,29
Pima	24,11	40,10	14,65	38,25
Wine	3,03	12,36	0,26	22,73
Ferrites	10,10	19,38	4,59	33,05
Remotes100	20,55	38,51	10,55	36,51
średnia	**17,36**	**33,46**	**10,03**	**34,94**



Rys. 7.9. Sumy rang poszczególnych metod na zbiorach UCI.
Mniejsze wartości są korzystniejsze.

Tabela 7.7b: Statystyki dotyczące klasyfikatora kaskadowego
voting k-NN + *voting k*-NSN (500 mutacji)

zbiór	**błąd ogółem [%]**	**% próbek w II etapie**	**błąd w I etapie [%]**	**błąd w II etapie [%]**
Bupa	30,96	50,96	25,86	34,93
Glass	26,46	34,11	17,32	43,27
Iris	4,53	5,73	2,98	32,14
Pima	24,11	19,19	20,29	40,00
Wine	2,14	5,84	0,47	30,80
Ferrites	9,88	13,84	4,75	42,16
Remotes100	22,88	26,03	15,56	43,67
Średnia	**17,28**	**22,24**	**12,46**	**38,14**

Analiza wyników zaprezentowanych w tym podrozdziale pozwala na wyciągnięcie kilku spostrzeżeń.

- Należy zauważyć, że średnio najniższą jakość klasyfikacji osiągnęła oryginalna reguła *k*-NN. Fakt ten z badawczego punktu widzenia oznacza, iż konstrukcja całkowicie nowych typów klasyfikatorów nie jest jałowym zajęciem.
- Niektóre klasyfikatory są mało stabilne na tle konkurencji. Przykładowo, klasyfikator *k*-NCN osiąga najlepszy wynik na zbiorze Bupa, natomiast najgorszy na zbiorze Glass. Praktyczną trudnością jest także odpowiedni wybór liczby mutacji dla klasyfikatora *k*-NSN (również w wersji z głosowaniem po wielu wartościach *k*).
- Wszystkie algorytmy z głosowaniem po wielu wartościach *k* osiągają zazwyczaj wyższą jakość klasyfikacji od swych pierwowzorów (np. *voting k*-NN jest przeważnie lepsza od oryginalnej *k*-NN itd.). Godny podkreślenia jest fakt, iż metody te są niewiele wolniejsze od wersji oryginalnych (w przypadku reguły *voting k*-NN, spowolnienie w stosunku do *k*-NN jest wręcz marginalne w typowym przypadku).
- Klasyfikatory kaskadowe cechują się korzystnym kompromisem między jakością a szybkością klasyfikacji. Jak wynika z Tabel 7.7, średnio ok. 70% predykcji próbek jest dokonywanych w pierwszym etapie (dla szybszego klasyfikatora pierwszego etapu, jakim jest sieć równoległa zbiorów zredukowanych Skalaka, odsetek ten jest nieco mniejszy, natomiast zbliża się do 80% dla klasyfikatorów korzystających z *voting k*-NN w pierwszej fazie). Wartości te są jednak mocno uzależnione od zbioru danych. Zauważyć trzeba, iż użyte kryterium podejmowania decyzji silnie dyskryminuje zbiory próbek testowych: ogólnie mówiąc, próbki „łatwe" (średnie błędy niewiele ponad 10%) są rozstrzygane w pierwszym etapie, próbki „trudne" (błędy ok. 35%–40%) obsługiwane są przez klasyfikator drugiego etapu. Własność ta jest korzystna, gdyż oznacza, iż klasyfikator może z sukcesem dobierać złożoność (tj. koszt czasowy) reguły decyzyjnej w zależności od stopnia trudności zadanej próbki.
- Dekompozycja zadania wielodecyzyjnego na sieć dychotomizerów wydaje się jedną z ważniejszych technik poprawy jakości klasyfikacji. Prawdopodobnie lepszym z dwóch przetestowanych schematów jest algorytm Moreiry–Mayoraza,

jednak silniejsze stwierdzenie wymaga przeprowadzenia większej liczby eksperymentów.

W najnowszej pracy autor rozprawy zaproponował jeszcze jeden klasyfikator kaskadowy (Grabowski, 2003b), tym razem oparty wyłącznie na zespołach zbiorach zredukowanych, z wykorzystaniem idei zawierania się zbiorów i jednomyślnego głosowania. Otrzymane wyniki pozwalają na przypuszczenie, że klasyfikator ten oferuje lepszy kompromis pomiędzy szybkością klasyfikacji a trafnością predykcji niż pojedynczy zespół zbiorów zredukowanych. Co więcej, algorytm ten można w naturalny sposób rozbudować dodając dalsze fazy klasyfikatora kaskadowego; wstępne wyniki (dotąd nieopublikowane) są bardzo obiecujące.

Dalsze plany badawcze autora obejmują także implementację i testy szereg innych hybryd kaskadowych, implementację wersji dekompozycyjnych rozważanych klasyfikatorów oraz większe zróżnicowanie decyzji dotyczącej dalszej klasyfikacji w zależności od odpowiedzi zespołu klasyfikatorów pierwszego etapu.

Podsumowanie i wnioski

W niniejszej rozprawie przedstawiono szereg nowych algorytmów klasyfikacji należących do rodziny klasyfikatorów minimalnoodległościowych, tj. wywodzących się z reguły k najbliższych sąsiadów (k-NN). Najważniejszymi kryteriami oceny przydatności klasyfikatora są jakość klasyfikacji (innymi słowy, trafność predykcji) oraz szybkość klasyfikacji. Ponieważ cele te wydają się przeciwstawne, uzasadnione wydawało się zaprojektowanie kilku algorytmów różniących się relacjami szybkości do trafności predykcji. Ponadto zaprezentowano nowy algorytm szybkiego deterministycznego szukania najbliższego sąsiada, który, rzecz jasna, nie jest klasyfikatorem, ale metodą umożliwiającą znajdowanie najbliższego sąsiada na potrzeby reguły decyzyjnej jeden najbliższy sąsiad w czasie subliniowym w liczności zbioru. W pracy pokazano ponadto, w jaki sposób można algorytm ten zaadoptować do znajdowania k sąsiadów scentrowanych, tj. sąsiadów wykorzystywanych w regule k-NCN oraz we wprowadzonej w niniejszej rozprawie regule k-NSN.

Ważną i dynamicznie w ostatnich latach rozwijaną koncepcją rozpoznawania obrazów są klasyfikatory o strukturze sieciowej (równoległe). Koncepcja ta jest również intensywnie wykorzystywana w niniejszej pracy. Autor rozprawy wprowadził ideę lokalnego wyboru zredukowanego zbioru odniesienia dla reguły 1-NN przy użyciu bardzo szybkiego kryterium selekcji zbioru. Zaproponowano koncepcję klasyfikatora *voting* k-NN będącego wersją reguły k-NN z głosowaniem po wielu wartościach parametru k. W odróżnieniu od większości klasyfikatorów równoległych z głosowaniem, spadek szybkości klasyfikacji w stosunku do klasyfikatora bazowego (tj. oryginalnej reguły k-NN) jest w zaproponowanym algorytmie w praktyce zaniedbywalny. Analogiczna idea została także wprowadzona i pomyślnie zweryfikowana dla innych klasyfikatorów typu k sąsiadów. W pracy badano również schematy kaskadowe i wprowadzono kryterium jednomyślnego głosowania zespołu komponentów w danym etapie klasyfikatora jako kryterium akceptacji predykcji. Kryterium to silnie dyskryminuje próbki „łatwe" od „trudnych" (zostało to ustalone w sposób eksperymentalny), co sugeruje jego przydatność w klasyfikatorach kaskadowych.

W niektórych zastosowaniach, np. zwykle w medycynie, jakość klasyfikacji ma zdecydowany priorytet. Jako „punkt odniesienia" w porównaniach jakości klasyfikacji

została w takich przypadkach przyjęta w pracy klasyczna reguła *k* najbliższych sąsiadów. W ostatnich latach pojawiła się interesująca koncepcja klasyfikatorów opartych na idei tzw. symetrycznego sąsiedztwa, które uwzględnia nie tylko bliskość sąsiadów, ale również ich układ geometryczny. W pracy zaproponowano nową regułę decyzyjną *k Near Surrounding Neighbors* (*k*-NSN), która optymalizuje oba kryteria używane przez jej poprzedniczkę, regułę *k* scentrowanych sąsiadów (*k*-NCN). Oba klasyfikatory oferują wyższą jakość klasyfikacji niż reguła *k*-NN, co zostało sprawdzone eksperymentalnie na szeregu rzeczywistych zbiorów, przy czym średnio mniejsze błędy popełniane były przez regułę *k*-NSN. Jakość predykcji oferowana przez te reguły została zbadana eksperymentalnie także w schematach dekompozycyjnych dla zadań wielodecyzyjnych (tj. dla których liczba klas jest większa niż 2). Oba wspomniane klasyfikatory są także intensywnie wykorzystywane w zaproponowanej przez autora rozprawy rodzinie klasyfikatorów kaskadowych. Algorytmy z tej rodziny osiągają jakość klasyfikacji niewiele niższą od najlepszych przetestowanych w pracy algorytmów minimalnoodległościowych przy co najmniej 2-3-krotnie wyższej szybkości.

Podsumowując, najważniejsze wnioski, jakie można wyciągnąć z przeprowadzonych badań są następujące:

1. W niskich wymiarach możliwe jest szukanie najbliższego sąsiada w czasie subliniowym w liczności zbioru w najgorszym przypadku; prezentowany algorytm dopuszcza użycie współczynnika kompromisu między szybkością szukania a kosztem wstępnej obróbki.
2. Lokalny wybór zredukowanego zbioru odniesienia oferuje wyższą jakość klasyfikacji niż klasyfikatory oparte na pojedynczym zbiorze zredukowanym (podejście klasyczne).
3. „Symetryczne sąsiedztwo" to nowy sposób poprawy jakości w rodzinie klasyfikatorów minimalnoodległościowych. Zaprezentowana reguła *k*-NSN optymalizuje oba kryteria używane w klasyfikatorze *k*-NCN.
4. Możliwa jest wersja reguły *k*-NN z wieloma wartościami *k* (wyższa jakość klasyfikacji za cenę minimalnego spowolnienia).
5. Koncepcje z p. 3 i 4 pozwalają na projektowanie klasyfikatorów kaskadowych o korzystnych relacjach szybkości do jakości klasyfikacji.

Wydaje się, że opracowane algorytmy mogą znaleźć liczne zastosowania m. in. w medycynie (analiza zdjęć, komputerowe wspomaganie diagnostyki), przemyśle (kontrola jakości) czy wojskowości (detekcja obiektów na zdjęciach lotniczych i satelitarnych).

Dalsze kierunki badań

Szereg aspektów zagadnień poruszonych w niniejszej pracy wymaga dalszych prac badawczych. W szczególności, warto rozpatrzyć następujące kwestie:

- dokonanie racjonalnego doboru parametrów dla schematu z lokalnym zbiorem odniesienia (metoda klasteryzacji, liczba klastrów, wielkość każdego zbioru zredukowanego);
- rozważenie zmiany strategii uczenia w algorytmie Skalaka. Używany oryginalnie algorytm genetyczny, z racji stosowania tylko jednego operatora genetycznego (mutacja), może mieć trudności z wyjściem z lokalnego minimum;
- implementacja „brakujących" połączeń opisywanych schematów z algorytmami dekompozycyjnymi dla zadań wielodecyzyjnych. Warto rozważyć użycie selekcji cech dla podzadań;
- powiększenie rodziny klasyfikatorów kaskadowych, zwłaszcza z wykorzystaniem zawierających się w sobie zbiorów zredukowanych; należy rozważyć użycie innych kryteriów wiarygodności predykcji;
- połączenie zaproponowanej wersji algorytmu MFS z kryterium oceny różnorodności zespołu rozważanych komponentów;
- analiza skuteczności techniki *voting* *k*-NN przy różnych licznościach zespołu komponentów i różnych metodach podziału zbioru uczącego na część konstrukcyjną i walidacyjną. Należy wziąć pod uwagę możliwy schemat z ważonym głosowaniem, np. w duchu idei Grossmana i Williamsa (1999) dla schematu *bagging*;
- sprawdzenie możliwości działania algorytmu *voting* *k*-NN z umiarkowanie zredukowanymi zbiorami odniesienia;
- pomiar jakości poszczególnych klasyfikatorów składowych w klasyfikatorach *k*-NN, *k*-NCN i *k*-NSN, a także korelacji między nimi. Przy założeniu adekwatności miary korelacji, może to rzucić nowe światło na pożądany kierunek konstrukcji dalszych klasyfikatorów wykorzystujących koncepcję symetrycznego sąsiedztwa.

Załącznik
Zbiory danych

Wszystkie używane w eksperymentach zbiory danych zawierały wyłącznie cechy numeryczne (ciągłe bądź dyskretne) i były zawsze kompletne, tj. nie zawierały brakujących wartości cech.

Zbiory UCI

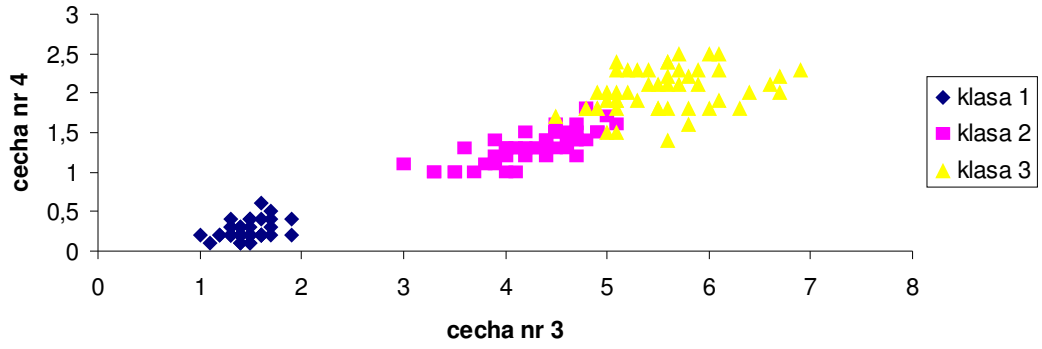
Zbiory te, należące do repozytorium Uniwersytetu Kalifornijskiego w Irvine (Machine Learning Repository, University of California, Irvine) (Merz i Murphy, 1996), są powszechnie wykorzystywane w literaturze przedmiotu.

Bupa — zbiór dotyczący wykrywania schorzeń wątroby w populacji męskiej związanych z nadużywaniem alkoholu. Pięć pierwszych cech to wyniki testów krwi, natomiast ostatnia cecha to liczba jednostek alkoholu przyjmowanych średnio w ciągu doby przez badanego mężczyznę.

Glass — zbiór próbek różnych rodzajów szkła (okienne, samochodowe *etc*.), identyfikowanych na podstawie zawartości określonych pierwiastków chemicznych (m. in. krzemu, sodu i wapnia). Zbiór zgromadzony przez kryminologów z Home Office Forensic Science Service w Reading w Wielkiej Brytanii.

Iris — zbiór próbek trzech podgatunków kosaćca, klasyfikowanych na podstawie czterech geometrycznych cech (długość i szerokość liścia oraz długość i szerokość płatka rośliny). Zbiór został spopularyzowany przez Fishera (1936).

Rzut zbioru na cechy 3 i 4 (najlepiej dyskryminujące klasy) został zaprezentowany na rysunku poniżej.



Rys. Zbiór Iris w rzucie dwuwymiarowym (cechy 3 i 4)

Pima — zbiór odnoszący się do zadania rozpoznania symptomów cukrzycy w oparciu o kryteria przyjęte przez Światową Organizację Zdrowia (WHO). Dane zostały zgromadzone na podstawie badań populacji Indianek z plemienia Pima (okolice Phoenix w Arizonie, USA).

Wine — zbiór dotyczący rozpoznania jednego z trzech gatunków win włoskich na podstawie cech wyekstrahowanych w wyniku analizy chemicznej.

Inne zbiory

Ferrites — zbiór dotyczący kontroli jakości rdzeni ferrytowych, które były produkowane w zakładach Polfer w Warszawie. Obraz danego rdzenia analizowany był piksel po pikslu, a zatem obiektami tworzącymi zbiór są pojedyncze piksle obrazu powierzchni rdzenia. Wyróżnione klasy stwierdzają, czy dany piksel należy do dobrej (nieuszkodzonej) części rdzenia, do tła, czy też do jednego z sześciu rodzajów defektów. Cechy opisujące każdy piksel wyekstrahowane są z jego sąsiedztwa (histogram jasności i momenty różnych stopni). Cechy zostały dobrane w taki sposób, aby ich wartości w niewielkim tylko stopniu zmieniały się przy obrotach danego rdzenia ferrytowego. Dokładny opis zbioru zawiera praca (Nieniewski i in., 1999).

FerritesOld — zbiór dotyczący tej samej aplikacji przemysłowej, co zbiór **Ferrites**, różniący się jednak od niego zestawem cech (wybrano zestaw mniejszy liczny), liczbą rozróżnianych klas i licznością.

RB8.PAT — kolejny zbiór dotyczący kontroli jakości rdzeni ferrytowych, charakteryzujący się znacznie większą liczbą cech (64) od zbiorów poprzednich. I tu cechy wydzielane były jednak z otaczającego dany piksel sąsiedztwa: maski o rozmiarach 9 na 9 piksli. Wyróżniono 14 klas: piksel należący do dobrej części powierzchni rdzenia, piksel należący do tła oraz do jednego z 12 rodzajów defektów (odpryski i pęknięcia w różny sposób usytuowane na rdzeniach).

Remotes — zbiór dotyczący detekcji obiektów (pól upraw) na zdjęciach lotniczych wykonanych w rejonie Feltwell w Wielkiej Brytanii. Rozróżnianymi klasami były: pole uprawne marchwi, ziemniaka, buraka cukrowego, pszenicy oraz ścierń. Cechy opisujące obiekty pozyskiwane były z dwóch sensorów: optycznego i radarowego. Zbiór ten opisany został bardziej szczegółowo w pracach (Roli, 1996) i (Grabowski i in., 2003). Dane niniejsze wykorzystywane były w grancie NATO nr PST.CLG.977258 (2001–2002) dotyczącym zastosowań nieparametrycznych metod rozpoznawania obrazów w aplikacjach *remote sensing*, którego kierownikiem był prof. C.-H. Chen z N. Dartmouth Coll., MA, USA, zaś współwykonawcami dr Adam Jóźwik i autor niniejszej rozprawy.

Ponadto w niektórych eksperymentach wykorzystano zbiory syntetyczne; są one opisane w tekście rozprawy (Rozdz. 5.2 i 5.4).

Poniżej zestawiono podstawowe wyznaczniki opisanych zbiorów rzeczywistych, tj. liczbę klas, cech i próbek.

zbiór danych	liczba klas	liczba cech	liczba próbek
Bupa	2	6	345
Glass	6	9	214
Iris	3	4	150
Pima	2	8	768
Wine	3	13	178
Ferrites	8	30	5904
FerritesOld	5	12	2885
RB8.PAT	14	64	4297
Remotes	5	9	5124

Bibliografia

1. AGARWAL, P.K., MATOUŠEK, J. (1992) Ray shooting and parametric search. 24th Annual ACM Symposium on Theory of Computing, Victoria, Canada, str. 517–526.
2. AHA, D.W. (1989) Incremental, instance-based learning of independent and graded concept descriptions. 6th International Workshop on Machine Learning, Ithaca, NY, USA, str. 387–391.
3. AHA, D.W., BANKERT, R.L. (1995) A comparative evaluation of sequential feature selection algorithms. 5th International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, str. 1–7.
4. AHA, D.W., GOLDSTONE, R.L. (1992) Concept learning and flexible weighting. 14th Annual Conference of the Cognitive Science Society, Evanston, IL, USA, str. 534–539.
5. AHO, A.V., HOPCROFT, J.E., ULLMAN, J.D. (2003) Projektowanie i analiza algorytmów. Wydawnictwo Helion, Gliwice. Rok pierwszego wydania oryginału: 1974.
6. AKKUS, A., GÜVENIR, H.A. (1996) K-nearest neighbor classification on feature projections. 13th International Conference on Machine Learning, Bari, Italy, str. 12–19.
7. ALMUALLIM, H., DIETTERICH, T.G. (1991) Learning with many irrelevant features. 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, str. 547–552.
8. ALPAYDIN, E. (1990) Neural Models of Incremental Supervised and Unsupervised Learning. PhD thesis, No 896, Department d'Informatique, Ecole Politechnique Fédérale de Lausanne, Switzerland.
9. ALPAYDIN, E. (1997) Voting over multiple condensed nearest neighbors. Artificial Intelligence Review, Vol. 11, No. 1–5, str. 115–132.
10. ALPAYDIN, E., JORDAN, M.I. (1996) Local linear perceptrons for classifycation. IEEE Transactions on Neural Networks, Vol. 7, No. 3, str. 788–792.
11. ALPAYDIN, E., KAYNAK, C. (1998) Cascading classifiers. Kybernetika, Vol. 34, No. 4, str. 369–374.
12. ANDERSEN, T., MARTINEZ, T.R. (1995) A provably convergent dynamic training method for multilayer perceptron networks. 2nd International Symposium on Neuroinformatics and Neurocomputers, Rostov-on-Don, Russia, str. 77–84.

13. ANDERSSON, A., DAVIDSSON, P., LINDÉN, J. (1999) Measure-based classifier performance evaluation. Pattern Recognition Letters, Vol. 20, No. 11–13, str. 1165–1173.

14. ATKESON, C.G., MOORE, A.W., SCHAAL, S. (1997) Locally weighted learning. Artificial Intelligence Review, Vol. 11, No. 1–5, str. 11–73.

15. BAILEY, T., JAIN, A.K. (1978) A note on distance-weighted k-nearest neighbor rules. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 4, str. 311–313.

16. BAIM, P. (1988) A method for attribute selection in inductive learning systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, str. 888–896.

17. BALKO, S., SCHMITT, I. (2002) Efficient nearest neighbor retrieval by using a local approximation technique — the active vertice approach. Preprint No. 2, Fakultät für Informatik, Universität Madgeburg, Germany.

18. BARAM, Y. (1998) Partial classification: The benefit of deferred decision. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, str. 769–776.

19. BAY, S.D. (1998) Combining nearest neighbour classifiers through multiple feature subsets. 15th International Conference on Machine Learning, Madison, WI, USA, str. 37–45.

20. BECKMANN, N., KRIEGEL, H.-P., SCHNEIDER, R., SEEGER, B. (1990) The R*-tree: An efficient and robust access method for points and rectangles. ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, USA, str. 322–331.

21. BENTLEY, J.L. (1975) Multidimensional binary search trees used for associative searching. Communications of the ACM, Vol. 18, No. 9, str. 509–517.

22. BENTLEY, J.L. (1979) Multidimensional binary search trees in database applications. IEEE Transactions on Software Engineering, Vol. 5, No. 5, str. 333–340.

23. BERCHTOLD, S., BÖHM, CH., JAGADISH, H.V., KRIEGEL, H.-P., SANDER, J. (2000) Independent quantization: An index compression technique for high-dimensional data spaces. 16th International Conference on Data Engineering (ICDE), San Diego, CA, USA, str. 577–588.

24. BEYER, K., GOLDSTEIN, J., RAMAKRISHNAN, R., SHAFT, U. (1999) When is „nearest neighbor" meaningful? 7th International Conference on Database Theory (ICDT), Jerusalem, Israel, str. 217–235.

25. BEZDEK, J.C., CHUAH, S.K., LEEP, D. (1986) Generalized k-nearest neighbor rules. Fuzzy sets and systems, Vol. 18, No. 2, str. 237–256.

26. BEZDEK, J.C., REICHHERZER, T.R., LIM, G.S., ATTIKIOUZEL, Y. (1998) Multiple-prototype classifier design. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-28, No. 1, str. 67–79.

27. BHATTACHARYYA, G.K., JOHNSON, R.A. (1977) Statistical Concepts and Methods. John Wiley and Sons, New York City, NY, USA.

28. BIENIECKI, W., **GRABOWSKI, SZ.**, SEKULSKA-NALEWAJKO, J., TURANT, M., KAŁUŻYŃSKI, A. (2002) System przetwarzania patomorfologicznych obrazów mikroskopowych. X Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 485–498.

29. BLANZIERI, E., RICCI, F. (1999) A minimum risk metric for nearest neighbour classification. Probability based metrics for nearest neighbour classification. 16th International Conference on Machine Learning, Bled, Slovenia, str. 22–31.

30. BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., WARMUTH, M. (1987) Occam's razor. Information Processing Letters, Vol. 24, No. 6, str. 377–380.

31. BORODIN, A., OSTROVSKY, R., RABANI, Y. (1999) Lower bounds for high dimensional nearest neighbor search and related problems. ACM Symposium on Theory of Computing (STOC), Atlanta, GA, USA, str. 312–321.

32. BOSE, R.C., RAY-CHAUDURI, D.K. (1960) On a class of error correcting binary group codes. Information and Control, Vol. 3, No. 1, str. 68–79.

33. BOTTOU, L., VAPNIK, V. (1992) Local learning algorithms. Neural Computation, Vol. 4, No. 6, str. 888–900.

34. BRANKE, J. (1995) Evolutionary algorithms for neural network design and training. 1st Nordic Workshop on Genetic Algorithms and its Applications, Vaasa, Finland, str. 145–163.

35. BREIMAN, L. (1996a) Bagging predictors. Machine Learning, Vol. 24, No. 2, str. 123–140.

36. BREIMAN, L. (1996b) Bias, variance and arcing classifiers. Technical Report TR 460, Dept. of Statistics, University of California, Berkeley, CA, USA.

37. BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A., STONE, C.J. (1984) Classification and Regression Trees. Wadsworth, Belmond, CA, USA.

38. BRILL, F.Z., BROWN, D.E., MARTIN, W.N. (1992) Fast genetic selection of features for neural network classifiers. IEEE Transactions on Neural Networks, Vol. 3, No. 2, str. 324–328.

39. BRIN, S. (1995) Near neighbor search in large metric spaces. 21st International Conference on Very Large Data Bases (VLDB), Zurich, Switzerland, str. 574–584.

40. BRODLEY, C.E. (1993) Addressing the selective superiority problem: Automatic algorithm / model class selection. 10th International Conference on Machine Learning, San Mateo, CA, USA, str. 17–24.

41. BROOMHEAD, D.S., LOWE, D. (1988) Multi-variable functional interpolation and adaptive networks. Complex Systems, Vol. 2, No. 3, str. 321–355.

42. BURKHARD, W.A., KELLER, R.M. (1973) Some approaches to best-match file searching. Communications of the ACM, Vol. 16, No. 4, str. 230–236.

43. CAMERON-JONES, R.M. (1995) Instance selection by encoding length heuristic with random mutation hill climbing. 8th Australian Joint Conference on Artificial Intelligence, Canberra, Australia, str. 99–106.

44. CARDIE, C. (1993) Using decision trees to improve case-based learning. 10th International Conference on Machine Learning, Amherst, MA, USA, str. 25–32.

45. CHANG, C.L. (1974) Finding prototypes for nearest neighbour classifiers. IEEE Transactions on Computers, Vol. 2–3, No. 11, str. 1179–1184.

46. CHAUDHURI, B.B. (1996) A new definition of neighbourhood of a point in multi-dimensional space. Pattern Recognition Letters, Vol. 17, No. 1, str. 11–17.

47. CHÁVEZ, E., NAVARRO, G., BAEZA-YATES, R., MARROQUÍN, J.L. (2001) Proximity searching in metric spaces. ACM Computing Surveys, Vol. 33, No. 3, str. 273–321.

48. CHÁVEZ, E., NAVARRO, G. (2001) Towards measuring the searching complexity of metric spaces. Encuentro Nacional de Computación (ENC'01), Vol. II, str. 969–978.

49. CHEN, C.-H., JÓŹWIK, A. (1996) A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recognition Letters, Vol. 17, No. 8, str. 819–823.

50. CHEN, S., COWAN, C.F.N., GRANT, P.M. (1991) Orthogonal least squares learning for radial basis function networks. IEEE Transactions on Neural Networks, Vol. 2, No. 2, str. 302–309.

51. CHERKAUER, K.J. (1996) Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. 13th AAAI Workshop on Integrated Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, USA, str. 15–21.

52. CHERKAUER, K.J., SHAVLIK, J.W. (1996) Growing simpler decision trees to facilitate knowledge discovery. 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, str. 315–318.

53. CIACCIA, P., PATELLA, M., ZEZULA, P. (1997) M-tree: An efficient access method for similarity search in metric spaces. 23rd International Conference on Very Large Data Bases (VLDB), Athens, Greece, str. 426–435.

54. CICHOSZ, P. (2000) Systemy uczące się. Wydawnictwa Naukowo-Techniczne, Warszawa.

55. CLARK, P., NIBLETT, T. (1989) The CN2 induction algorithm. Machine Learning, Vol. 3, No. 4, str. 261–283.

56. CLARKSON, K.L. (1988) A randomised algorithm for closest-point queries. SIAM Journal on Computing, Vol. 17, No. 4, str. 830–847.

57. CLARKSON, K.L. (1999) Nearest neighbor queries in metric spaces. Discrete Computational Geometry, Vol. 22, No. 1, str. 63–93.

58. COGNITIVE SYSTEMS, INC. (1992) ReMind: Case-based Reasoning Development Shell.

59. COST, S., SALZBERG, S. (1993) A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning, Vol. 10, No. 1, str. 57–78.

60. COVER, T.M., HART, P.E. (1967) Nearest neighbour pattern classification. IEEE Transactions on Information Theory, Vol. IT-13, No. 1, str. 21–27.

61. CREECY, R.H., MASAND, B.M., SMITH, S.J., WALTZ, D.L. (1992) Trading MIPS and memory for knowledge engineering. Communication of the ACM, Vol. 35, No. 8, str. 48–63.

62. DASARATHY, B.V. (ed.) (1991) Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos, CA, USA.

63. DASARATHY, B.V., SHEELA, B.V. (1979) A composite classifier system design: concepts and methodology. Proceedings of the IEEE (Special Issue on Pattern Recognition and Image Processing), Vol. 67, No. 5, str. 708–713.

64. DIETTERICH, T.G., BAKIRI, G. (1991) Error-correcting output codes: A general method of improving multiclass inductive learning programs. 9th National Conference on Artificial Intelligence (AAA-91), Anaheim, CA, USA, str. 572–577.

65. DIETTERICH, T.G., BAKIRI, G. (1995) Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, Vol. 2, str. 263–286.

66. DIXON, J.K. (1979) Pattern recognition with partly missing data. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9, No. 10, str. 617–621.

67. DOAK, J. (1992) An evaluation of feature selection methods and their application to computer security. Technical Report CSE-92-18, Dept. of Computer Science, University of California, Davis, CA, USA.

68. DOBKIN, D., LIPTON, R. (1976) Multidimensional search problems. SIAM Journal on Computing, Vol. 5, No. 2, str. 181–186.

69. DOMINGOS, P. (1997) Context-sensitive feature selection for lazy learners. Artificial Intelligence Review, Vol. 11, No. 1–5, str. 227–253.

70. DOMINGOS, P. (1998) Occam's two razors: the sharp and the blunt. 4th International Conference on Knowledge Discovery and Data Mining, New York City, NY, USA, str. 37–43.

71. DOMINGOS, P., PAZZANI, M. (1996) Beyond independence: Conditions for the optimality of the simple Bayesian classifier. 13th International Conference on Machine Learning, Bari, Italy, str. 105–112.

72. DOUGHERTY, J., KOHAVI, R., SAHAMI, M. (1995) Supervised and unsupervised discretization of continuous features. 12th International Conference on Machine Learning, Tahoe City, CA, USA, str. 194–202.

73. DUDA, R.O., HART, P.E. (1973) Pattern Classification and Scene Analysis. John Wiley and Sons, New York City, NY, USA.

74. DUDANI, S.A. (1976) The distance-weighted k-nearest neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, No. 4, str. 325–327.

75. FALHMAN, S.E. (1988) Faster-learning variations on backpropagation: An empirical study. Connectionist Models Summer School, str. 38–51.

76. FENG, C., KING, R., SUTHERLAND, A. (1993) Statlog: Comparison of machine learning, statistical and neural network classification algorithms. Technical report, The Turing Institute.

77. FISHER, R.A. (1936) The use of multiple measurements in taxonomical problems. Annals of Eugenics, Vol. 7, str. 179–188.

78. FIX, E., HODGES JR., J.L. (1951) Discriminatory analysis — nonparametric discrimination: Consistency properties. Project 21-49-004, Report No. 4, USAF School of Aviation Medicine, Randolph Field, TX, USA, str. 261–279.

79. FIX, E., HODGES JR., J.L. (1952) Discriminatory analysis — nonparametric discrimination: Small sample performance. Project 21-49-004, Report No. 11, USAF School of Aviation Medicine, Randolph Field, TX, USA, str. 280–322.

80. FREUND, Y., SCHAPIRE, R.E. (1996) Experiments with a new boosting algorithm. 13th International Conference on Machine Learning, Bari, Italy, str. 148–156.

81. FRIEDRICH, CH.M. (1998) Ensembles of evolutionary created artificial neural networks. 3rd On-line Conference on Soft Computing in Engineering Design and Manufacturing (WSC3). W: R. Roy, T. Furuhashi & P.K. Chawdhry, (eds.), Advances in Soft Computing, Springer, str. 288–298, URL: http://www.tussy.uni-wh.de/~chris/publications/friedrich.wsc3.pdf.

82. FRIEDMAN, J.H. (1994) Flexible metric nearest neighbour classification. Technical report, Dept. of Statistics, Stanford University, CA, USA, 1994.

83. FRIEDMAN, J.H., BENTLEY, J.L., FINKEL, R.A. (1977) An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, Vol. 3, str. 209–226.

84. FRIEDMAN, J.H., KOHAVI, R., YUN, Y. (1996) Lazy decision trees. 13th National Conference on Artificial Intelligence, Portland, OR, USA, str. 717–724.

85. FU, K.S. (1968) Sequential methods in pattern recognition and machine learning. Academic Press, New York City, NY, USA.

86. FUKUNAGA, K. (1990) Introduction to Statistical Pattern Recognition (Second Edition). Academic Press, New York City, NY, USA.

87. GAMA, J.M.P. DA (1999) Combining Classification Algorithms. PhD thesis, Dept. of Computer Science, University of Porto, Porto, Portugal.

88. GATES, G.W. (1972) The reduced nearest neighbor rule. IEEE Transactions on Information Theory, Vol. IT-18, No. 5, str. 431–433.

89. GIACINTO, G., ROLI, F. (1997) Adaptive selection of image classifiers. 9th International Conference on Image Analysis and Processing, Florence, Italy, str. 38–45.

90. GOWDA, K.C., KRISHNA, G. (1979) The condensed nearest neighbour rule using the concept of mutual nearest neighbourhood. IEEE Transactions on Information Theory, Vol. IT-25, No. 4, str. 488–490.

91. **GRABOWSKI, SZ.** (1999) Szybkie algorytmy redukcji zbioru odniesienia dla klasyfikatora typu 1-NS. VII Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 241–250.

92. **GRABOWSKI, SZ.** (2000a) Modyfikacje algorytmów redukcji zbiorów odniesienia dla klasyfikatora typu najbliższy sąsiad. VIII Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 381–398.

93. **GRABOWSKI, SZ.** (2000b) Porównanie metod selekcji cech dla klasyfikatorów minimalnoodległościowych. VIII Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 399–409.

94. **GRABOWSKI, SZ.** (2001a) Fast deterministic exact nearest neighbor search in the Manhattan metric. II Konferencja „Komputerowe Systemy Rozpoznawania" (KOSYR 2001), Miłków k/Karpacza, str. 375–379.

95. **GRABOWSKI, SZ.** (2001b) Experiments with the k-NCN decision rule. IX Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 307–317.

96. **GRABOWSKI, SZ.** (2002a) Selecting subsets of features for the MFS classifier via a random mutation hill climbing technique. International IEEE Conference TCSET'2002, Lviv–Slavskie, Ukraine, str. 221–222.

97. **GRABOWSKI, SZ.** (2002b) Voting over multiple k-NN classifiers. International IEEE Conference TCSET'2002, Lviv–Slavskie, Ukraine, str. 223–225.

98. **GRABOWSKI, SZ.** (2002c) Lokalny wybór zredukowanego zbioru odniesienia. Seminarium nt. „Przetwarzanie i analiza sygnałów w systemach wizji i sterowania", Słok k/Bełchatowa, str. 142–147.

99. **GRABOWSKI, SZ.** (2002d) Towards decision rule based on closer symmetric neighborhood. Biocybernetics and Biomedical Engineering, kwartalnik PAN, przyjęte do druku.

100. **GRABOWSKI, SZ.** (2003a) A family of cascade NN-like classifiers. 7th International IEEE Conference on Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv–Slavske, Ukraine, str. 503–506.

101. **GRABOWSKI, SZ.** (2003b) Telescope ensembles of reduced sets. III Konferencja „Komputerowe Systemy Rozpoznawania" (KOSYR 2003), Miłków k/Karpacza, praca przyjęta.

102. **GRABOWSKI, SZ.**, BARANOWSKI, M. (2002) Implementacja algorytmu szybkiego deterministycznego szukania najbliższego sąsiada w metryce miejskiej. X Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 499–514.

103. **GRABOWSKI, SZ.**, JÓŹWIK, A. (1999) Redukcja edytowanego zbioru odniesienia dla klasyfikatora typu 1-NS przy użyciu poprawionego algorytmu redukcji Tomeka. VII Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 219–227.

104. **GRABOWSKI, SZ.**, JÓŹWIK, A. (2003) Sample set reduction for nearest neighbor classifiers under different speed requirements. 7th International IEEE Conference on Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv–Slavske, Ukraine, str. 465–468.

105. **GRABOWSKI, SZ.**, JÓŹWIK, A., CHEN, C.-H. (2003) Nearest neighbor decision rule for pixel classification in remote sensing. Praca nadesłana jako część monografii „Frontiers of Remote Sensing Info Processing", ed. Steven Patt, World Scientific Publishing Co. Pte. Ltd., Republic of Singapore.

106. **GRABOWSKI, SZ.**, SOKOŁOWSKA, B. (2003) Voting over multiple k-NN and k-NCN classifiers for detection of respiration pathology. III Konferencja „Komputerowe Systemy Rozpoznawania" (KOSYR 2003), Miłków k/Karpacza, praca przyjęta.

107. GROSSMAN, D., WILLIAMS, T. (1999) Machine learning ensembles: An empirical study and novel approach. Unpublished manuscript, University of Washington, Seattle, WA, USA, URL: http://www.cs.washington.edu/homes/#grossman/projects/573projects/learning.

108. GUO, Z., UHRIG, R.E. (1992) Using genetic algorithms to select inputs for neural networks. International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92), Los Alamitos, NM, USA, str. 223–234.

109. GUTTMAN, A. (1984) R-trees: A dynamic index structure for spatial searching. ACM SIGMOD International Conference on Management of Data, Boston, MA, USA, str. 47–57.

110. HALL, M. (1995) Selection of attributes for modeling Bach chorales by a genetic algorithm. 2nd New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, Dunedin, New Zealand, str. 182–185.

111. HANSEN, L.K., SALAMON, P. (1990) Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, str. 993–1001.

112. HART, P.E. (1968) The condensed nearest neighbour rule. IEEE Transactions on Information Theory, Vol. IT-14, No. 3, str. 515–516.

113. HASTIE, T., TIBSHIRANI, R. (1996) Discriminant adaptive nearest neighbor classification. IEEE Pattern Analysis and Machine Intelligence, Vol. 18, No. 6, str. 607–616.

114. HEBB, D.O. (1949) The Organization of Behavior: A Neuropsychological Theory. John Wiley and Sons, New York City, NY, USA.

115. HECHT-NIELSEN, R. (1987) Counterpropagation networks. Applied Optics, Vol. 26, No. 23, str. 4979–4984.

116. HO, T.K. (1998a) The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, str. 832–844.

117. HO, T.K. (1998b) Nearest neighbors in random subspaces. 2nd International Workshop on Statistical Techniques in Pattern Recognition, Sydney, Australia, str. 640–648.

118. HOLMES, G., NEVILL-MANNING, C.G. (1995) Feature selection via the discovery of simple classification rules. Symposium on Intelligent Data Analysis, Baden-Baden, Germany.

119. HOLTE, R.C. (1993) Very simple classification rules perform well on most commonly used datasets. Machine Learning, Vol. 11, No. 1, str. 63–91.

120. HUANG, Y.S., SUEN, C.Y. (1995) A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 1, str. 90–93.

121. HYAFIL, L., RIVEST, R.L. (1976) Constructing optimal binary decision trees is NP-complete. Information Processing Letters, Vol. 5, No. 1, str. 229–246.

122. INDURKHYA, N., WEISS, S.M. (1998) Estimating performance gains for voted decision trees. Intelligent Data Analysis, Vol. 2, No. 4.

123. INDYK, P., MOTWANI, R. (1998) Approximate nearest neighbors: Towards removing the curse of dimensionality. 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, str. 604–613.

124. JACOBS, R.A., JORDAN, M.I., NOWLAN, S.J., HINTON, G.E. (1991) Adaptive mixtures of local experts. Neural Computation, Vol. 3, No. 1, str. 79–87.

125. JAIN, A.K., DUIN, R.P.W., MAO, J. (2000) Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, str. 4–37.

126. JAJUGA, K. (1990) Statystyczna teoria rozpoznawania obrazów. PWN, Warszawa.

127. JANKOWSKI, N. (1999) Ontogeniczne sieci neuronowe w zastosowaniu do klasyfikacji danych medycznych. Rozprawa doktorska, Katedra Metod Komputerowych, Uniwersytet Mikołaja Kopernika, Toruń.

128. JAROMCZYK, J.W., TOUSSAINT, G.T. (1992) Relative neighbourhood graphs and their relatives. Proc. IEEE 80, str. 1502–1517.

129. JOHN, G., KOHAVI, R., PFLEGER, K. (1994) Irrelevant features and the subset selection problem. 11th International Conference on Machine Learning, New Brunswick, NJ, USA, str. 121–129.

130. JÓŹWIK, A. (1983) A learning scheme for a fuzzy k-NN rule. Pattern Recognition Letters, Vol. 1, No. 5–6, str. 287–289.

131. JÓŹWIK, A. (2001) Porównanie klasyfikatora standardowego i równoległej sieci dwudecyzyjnych klasyfikatorów typu k-NS. IX Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 381–389.

132. JÓŹWIK, A., CHMIELEWSKI, L., CUDNY, W., SKŁODOWSKI, M. (1996) A 1-NN preclassifier for fuzzy k-NN rule. 13th International Conference on Pattern Recognition, Vienna, Austria, Vol. IV, track D, str. 234–238.

133. JÓŹWIK, A., **GRABOWSKI, SZ.** (2003) Metody konstrukcji szybkich klasyfikatorów do analizy obrazów optycznych. Prace Przemysłowego Instytutu Elektroniki (PAN), praca nadesłana.

134. JÓŹWIK, A., SERPICO, S.B., ROLI, F. (1995) Condensed version of the k-NN rule remote sensing image classification. Image and Signal Processing for Remote Sensing II, EUROPTO Proceedings, SPIE, Vol. 2579, Paris, France, str. 196–198.

135. JÓŹWIK, A., STAWSKA, Z. (1999) Wykorzystanie reguł 1-NS i k-NS oraz metody k średnich do konstrukcji szybkiego klasyfikatora. VII Konferencja „Sieci i Systemy Informatyczne: teoria, projekty, wdrożenia, aplikacje", Łódź, str. 241–250.

136. JÓŹWIK, A., VERNAZZA, G. (1988) Recognition of leucocytes by a parallel k-NN classifiers. Lecture Notes of ICB Seminar, Warszawa, str. 138–153.

137. KALANTARI, I., MCDONALD, G. (1983) A data structure and an algorithm for the nearest point problem. IEEE Transactions on Software Engineering, Vol. 9, No. 5, str. 631–634.

138. KATAYAMA, N., SATOH, S. (1997) The SR-tree: An index structure for high dimensional nearest neighbor queries. ACM SIGMOD International Conference on Management of Data, Tucson, AZ, USA, str. 369–380.

139. KAYNAK, C., ALPAYDIN, E. (2000) Multistage cascading of multiple classifiers: One man's noise is another man's data. 17th International Conference on Machine Learning, Stanford University, CA, USA, str. 455–462.

140. KELLER, J.M., GRAY, M.R., GIVENS JR., J.A. (1985) A fuzzy k-nearest neighbor algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, No. 4, str. 580–585.

141. KELLY, J.D., DAVIS, L. (1991) A hybrid genetic algorithm for classification. 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, str. 645–650.

142. KIBLER, D., AHA, D.W. (1987) Learning representative exemplars of concept: An initial case study. 4th International Workshop on Machine Learning, Irvine, CA, USA, str. 24–30.

143. KIRA, K., RENDELL, L.A. (1992) A practical approach to feature selection. 9th International Conference on Machine Learning, Aberdeen, Scotland, str. 249–256.

144. KITTLER, J. (1978) Feature set search algorithms. W: C.-H. Chen (ed.), Pattern Recognition and Signal Processing, Sijhoff am Noordhoff, the Netherlands.

145. KITTLER, J., HATEF, M., DUIN, R.P.W., MATAS, J. (1998) On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, str. 226–239.

146. KNUTH, D.E. (1973) The Art of Computer Programming: Sorting and Searching. Addison-Wesley, Reading, MA, USA.

147. KOHAVI, R. (1996) Scaling up the accuracy of naïve-Bayes classifiers: A decision-tree hybrid. 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, str. 202–207.

148. KOHAVI, R., WOLPERT, D.H. (1996) Bias plus variance decomposition for zero-one loss functions. 13th International Conference on Machine Learning, Bari, Italy, str. 275–283.

149. KOHONEN, T. (1986) Learning vector quantization for pattern recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland.

150. KOLLEN, J.F., POLLACK, J.B. (1991) Back propagation is sensitive to initial conditions. W: Advances in Neural Information Processing Systems, Morgan Kaufmann, Vol. 3, San Francisco, CA, USA, str. 860–867.

151. KOLLER, D., SAHAMI, M. (1996) Toward optimal feature selection. 13th International Conference on Machine Learning, Bari, Italy, str. 284–292.

152. KONONENKO, I. (1991) Semi-naïve bayesian classifier. 6th European Working Session on Learning, Porto, Portugal, str. 206–219.

153. KRUSKAL JR., J.B. (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematics Society, Vol. 7, No. 1, str. 48–50.

154. KUBAT, M., CHEN, W.K. (1998) Weighted projection in nearest-neighbor classifiers. 1st Southern Symposium on Computation, Hattiesburg, MS, USA.

155. KUDO, M., SKLANSKY, J. (2000) Comparison of algorithms that select features for pattern classifiers. Pattern Recognition, Vol. 33, No. 1, str. 25–41.

156. KUNCHEVA, L.I. (1995) Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Letters, Special Issue on Genetic Algorithms, Vol. 16, No. 8, str. 809–814.

157. KUNCHEVA, L.I. (2000) Cluster-and-selection method for classifier combination. 4th International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies (KES'2000), Brighton, UK, str. 185–188.

158. KUNCHEVA, L.I. (2001) Reducing the computational demand of the nearest neighbor classifier. School of Informatics, Symposium on Computing 2001, Aberystwyth, UK, str. 61–64.

159. KUNCHEVA, L.I. (2002) Switching between selection and fusion in combining classifiers: An experiment. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-32, No. 2, str. 146–156.

160. KUNCHEVA, L.I., BEZDEK, J.C., DUIN, R.P.W. (2001) Decision templates for multiple classifier fusion. Pattern Recognition, Vol. 34, No. 2, str. 299–314.

161. KUNCHEVA, L.I., JAIN, L.C. (1999) Nearest neighbor classifier: Simultaneous editing and feature selection. Pattern Recognition Letters, Vol. 20, No. 11–13, str. 1149–1156.

162. KUNCHEVA, L.I., WHITAKER, C.J., SHIPP, C.A., DUIN, R.P.W. (2000) Is independence good for combining classifiers? 15th International Conference on Pattern Recognition, Barcelona, Spain, str. 168–171.

163. KUNCHEVA, L.I., WHITAKER, C.J. (2003) Measures of diversity in classifier ensembles. Machine Learning, Vol. 51, No. 2, str. 181–207.

164. KURZYŃSKI, M. (1997) Rozpoznawanie obiektów — metody statystyczne. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław.

165. KUSHILEVITZ, E., OSTROVSKY, R., RABANI, Y. (1998) Efficient search for approximate nearest neighbor in high dimensional spaces. 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, str. 614–623.

166. LANGLEY, P. (1993) Induction of recursive Bayesian classifiers. 8th European Conference on Machine Learning, Vienna, Austria, str. 153–164.

167. LANGLEY, P., IBA, W. (1993) Average-case analysis of a nearest neighbor algorithm. 13th International Joint Conference on Artificial Intelligence, Chambery, France, str. 889–894.

168. LANGLEY, P., SAGE, S. (1994) Induction of selective Bayesian classifiers. 10th Conference on Uncertainty in Artificial Intelligence, Seattle, WA, USA, str. 399–406.

169. LEE, C. (1994) An instance-based learning method for databases: An information theoretic approach. 9th European Conference on Machine Learning, Catania, Italy, str. 387–390.

170. LEE, R.C.T., SLAGLE, J.R., MONG, C.T. (1976) Application of clustering to estimate missing data and improve data integrity. 2nd International Conference on Software Engineering, San Francisco, CA, USA, str. 539–544.

171. LIAO, S., LOPEZ, M.A., LEUTENEGGER, S.T. (2001) High dimensional similarity search with space filling curves. International Conference on Data Engineering, Heidelberg, Germany, str. 615–622.

172. LOWE, D.G. (1995) Similarity metric learning for a variable-kernel classifier. Neural Computation, Vol. 7, No. 1, str. 72–85.

173. MACQUEEN, J. (1967) Some methods for classification and analysis of multivariate observations. 5th Berkeley Symposium on Mathematics, Statistics and Probability, University of California Press, Berkeley, CA, USA, Vol. 1, str. 281–296.

174. MARON, O., MOORE, A.W. (1997) The racing algorithm: Model selection for lazy learners. Artificial Intelligence Review, Vol. 11, No. 1–5, str. 193–225.

175. MARQUIS CONDORCET, J.A. (1781) Sur les elections par scrutiny. Historie de l'Academie Royale des Sciences, str. 31–34.

176. MARILL, T., GREEN, D.M. (1963) On the effectiveness of receptors in recognition systems, IEEE Transactions on Information Theory, Vol. IT-9, No. 1, str. 1–17.

177. MARTIN, J.K. (1997) An exact probability metric for decision tree splitting and stopping. Machine Learning, Vol. 28, No. 2–3, str. 257–291.

178. MASULLI, F., VALENTINI, G. (2000) Comparing decomposition methods for classification. 4th International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, Piscataway, NJ, USA, str. 788–791.

179. MATHEUS, C.J., RENDELL, L.A. (1989) Constructive induction on decision trees. 11th International Joint Conference on Artificial Intelligence, Detroit, MI, USA, str. 645–650.

180. MATOUŠEK, J. (1992) Reporting points in halfspaces. Computation Geometry: Theory and Applications, Vol. 2, No. 3, str. 169–186.

181. MCCULLOCH, W.S., PITTS, W.H. (1943) A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, Vol. 5, str. 115–133.

182. MEISER, S. (1993) Point location in arrangements of hyperplanes. Information and Computation, Vol. 106, No. 2, str. 286–303.

183. MERZ, CH., MURPHY, P.M. (1996) UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

184. MICHALSKI, R.S., STEPP, R.E., DIDAY, E. (1981) A recent advance in data analysis: clustering objects into classes characterized by conjuctive concepts. W: L.N. Kanal & A. Rozenfeld (eds.), Progress in Pattern Recognition, Vol. 1, New York: North-Holland, USA, str. 33–56.

185. MICHIE, D., SPIEGELHALTER, D.J., TAYLOR, C.C. (1994) Machine Learning, Neural and Statistical Classification. Ellis Horwood, New York City, NY, USA.

186. MINSKY, M.L., PAPERT, S. (1969) Perceptrons: An introduction to computational geometry. MIT Press, Cambridge, MA, USA.

187. MITCHELL, T. (1980) The need for biases in learning generalizations. Technical Report, Computer Science Dept., Rutgers University, New Brunswick, NJ, USA. W: J.W. Shavlik & T. Dietterich, eds. (1990), Readings in Machine Learning, Morgan Kaufmann, San Mateo, CA, USA, str. 184–191.

188. MITCHELL, T. (1997) Machine Learning. MacGrew-Hill Companies, Inc., New York City, NY, USA.

189. MOHRI, T., TANAKA, H. (1994) An optimal weighting criterion of case indexing for both numeric and symbolic attributes. AAAI Workshop on Case-Based Reasoning, Seattle, WA, USA, str. 123–127.

190. MOLLINEDA, R.A., FERRI, F., VIDAL, E. (2000) Merge-based prototype selection for nearest neighbor classification. 4th World Multiconference on Systemics, Cybernetics and Informatics, Vol. VII (Computer Science and Enginnering), Orlando, FL, USA, str. 640–645.

191. MOODY, J., DARKEN, C.J. (1989) Fast learning in networks of locally-tuned processing units. Neural Computation, Vol. 1, No. 2, str. 281–294.

192. MOREIRA, M., MAYORAZ, E. (1998) Improved pairwise coupling classification with correcting classifiers. 10th European Conference on Machine Learning (ECML), Chemnitz, Germany, str. 160–171.

193. MORIN, R.L., RAESIDE, D.E. (1981) A reappraisal of distance-weighted k-nearest neighbor classification for pattern recognition with missing data. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-11, No. 3, str. 241–243.

194. MUCCIARDI, A.N., GOSE, E.E. (1971) A comparison of seven techniques for choosing subsets of pattern recognition properties. IEEE Transactions on Computers, Vol. C-20, No. 9, str. 1023–1031.

195. MURTAGH, F. (1985) Multidimensional Clustering Algorithms. Physica-Verlag, Vienna, Austria, 1985.

196. NARENDRA, P.M., FUKUNAGA, K. (1977) A branch and bound algorithm for feature subset selection. IEEE Transactions on Computers, Vol. C-26, No. 9, str. 917–922.

197. NIENIEWSKI, M., CHMIELEWSKI, L., JÓŹWIK, A., SKŁODOWSKI, M. (1999) Morphological detection and feature-based classification of cracked regions in ferrites, Machine Graphics and Vision, Vol. 8, No. 4.

198. NILSSON, N.J. (1965) Learning machines: Foundations of trainable pattern-classifying systems. McGraw Hill, New York City, NY, USA.

199. NOSOFSKY, R.M., CLARK, S.E., SHIN, H.J. (1989) Rules and exemplars in categorization, identification, and recognition. Journal of Experimental Psychology: Learning, Memory, and Cognition, Vol. 15, No. 2, str. 282–304.

200. O'CALLAGHAN, J.F. (1975) An alternative definition for "neighborhood of a point". IEEE Transactions on Computing, Vol. 24, No. 1, str. 1121–1125.

201. OPITZ, D., MACLIN, R. (1999) Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, Vol. 11, str. 169–198.

202. PAGEL, B.-U., KORN, F., FALOUTSOS, CH. (2000) Deflating the dimensionality curse using multiple fractal dimensions. 16th International Conference on Data Engineering (ICDE), San Diego, CA, USA, str. 589–598.

203. PARMANTO, B., MUNRO, P.W., DOYLE, H.R. (1996) Improving committee diagnosis with resampling techniques. W: D.S. Touretzky, M.C. Mozer & M.E. Hesselmo (eds.), Advances in Neural Information Processing Systems, Cambridge, MA, USA, MIT Press, Vol. 8, str. 882–888.

204. PAZZANI, M. (1996) Constructive induction of Cartesian product attributes. Conference ISIS96: Information, Statistics and Induction in Science, Melbourne, Australia, str. 66–77.

205. PRECHELT, L. (1996) A quantative study of experimental evaluations of neural network learning algorithms: Current research practice. Neural Networks, Vol. 9, No. 3, str. 457–462.

206. PREPARATA, F.P., SHAMOS, M.I. (1985) Computational Geometry: An Introduction. Springer-Verlag, New York City, NY, USA.

207. QUINLAN, J.R. (1979) Discovering rules from large collections of examples: a case study. W: Expert Systems in the Micro-electronic Age, Edinburgh University Press, Edinburgh, UK, str. 168–201.

208. QUINLAN, J.R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, USA.

209. RASTRIGIN, L.A., ERENSTEIN, R.H. (1981) Method of Collective Recognition. Energoizdat, Moscow, SU (w języku rosyjskim).

210. RAUBER, T.W., STEIGER-GARÇÃO, A.S. (1993) Feature selection of categorical attributes based on contingency table analysis. 5th Portuguese Conference on Pattern Recognition, Porto, Portugal.

211. RAVIV, Y., INTRATOR, N. (1996) Bootstrapping with noise: an effective regularization technique. Connection Science, Vol. 8, No. 3–4, str. 355–372.

212. RICCI, F., AHA, D.W. (1998) Error-correcting output codes for local learners. 10th European Conference on Machine Learning, Chemnitz, Germany, str. 280–291.

213. RICCI, F., AVESANI, P. (1995) Learning a local similarity metric for case-based reasoning. International Conference on Case-Based Reasoning (ICCBR), Sesimbra, Portugal, str. 301–312.

214. RIEDMILLER, M., BRAUN, H. (1993) A direct adaptive method for faster backpropagation learning: The RPROP algorithm. IEEE Conference on Neural Networks, San Francisco, CA, USA, str. 586–591.

215. RIMER, M.E., MARTINEZ, T.R., WILSON, D.R. (2002) Improving speech recognition learning through lazy learning. IEEE International Joint Conference on Neural Networks (IJCNN'02), Honolulu, HI, USA, str. 2568–2573.

216. RITTER, G.L., WOODROOF, H.B., LOWRY, S.R., ISENHOUR, T.L. (1975) An algorithm for a selective nearest neighbour decision rule. IEEE Transactions on Information Theory, Vol. IT-21, No. 6, str. 665–669.

217. ROLI, F. (1996) Multisensor image recognition by neural networks with understandable behaviour. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 10, No. 8, str. 887–917.

218. ROSENBLATT, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, Vol. 65, str. 386–408.

219. RUMELHART, D.E., HINTON, G.E., WILLIAMS, R.J. (1986) Learning internal representations by error propagation. W: D.E. Rumelhart, & J.L. McClelland, eds. (1986), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, str. 318–362, MIT Press, Cambridge, MA, USA.

220. RUTKOWSKA, D., PILIŃSKI, M., RUTKOWSKI, L. (1997) Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. PWN, Warszawa–Łódź.

221. SALZBERG, S. (1991) A nearest hyperrectangle learning method. Machine Learning, Vol. 6, No. 3, str. 251–276.

222. SÁNCHEZ, J.S., PLA, F., FERRI, F.J. (1997a) Prototype selection for the nearest neighbour rule through proximity graphs. Pattern Recognition Letters, Vol. 18, No. 6, str. 507–513.

223. SÁNCHEZ, J.S., PLA, F., FERRI, F.J. (1997b) On the use of neighbourhood-based non-parametric classifiers. Pattern Recognition Letters, Vol. 18, No. 11–13, str. 1179–1186.

224. SÁNCHEZ, J.S., PLA, F., FERRI, F.J. (1998) Improving the k-NCN classification rule through heuristic modifications. Pattern Recognition Letters, Vol. 19, No. 13, str. 1165–1170.

225. SCHAFFER, C. (1994) A conservation law for generalization performance. 11th International Conference on Machine Learning, New Brunswick, NJ, USA, str. 259–265.

226. SCHAPIRE, R.E. (1990) The strength of weak learnability. Machine Learning, Vol. 5, No. 2, str. 197–227.

227. SCHAPIRE, R.E., FREUND, Y., BARTLETT, P., LEE, W.S. (1997) Boosting the margin: A new explanation for the effectiveness of voting methods. 14th International Conference on Machine Learning, Nashville, TN, USA.

228. SCHIFFMANN, W., JOOST, M., WERNER, R. (1993) Comparison of optimized backpropagation algorithms. European Symposium on Artificial Neural Networks (ESANN), Brussels, Belgium, str. 97–104.

229. SHEPHERD, J., ZHU, X., MEGIDDO, N. (1999) A fast indexing method for multidimensional nearest neighbor search. SPIE Conference on Storage and Retrieval of Image and Video Databases, San Jose, CA, USA, str. 350–355.

230. SHIPP, C.A., KUNCHEVA, L.I. (2002) Relationships between combination methods and measures of diversity in combining classifiers. Information Fusion, Vol. 3, No. 2, str. 135–148.

231. SHORT, R.D., FUKUNAGA, K. (1980) A new nearest neighbour distance measure. 5th IEEE International Conference on Pattern Recognition, Miami Beach, FL, USA, str. 81–86.

232. SHORT, R.D., FUKUNAGA, K. (1981) The optimal distance measure for nearest neighbour classification. IEEE Transactions on Information Theory, Vol. IT-27, No. 5, str. 622–627.

233. SIERRA, B., LARRAÑAGA, P., INZA, I. (2000) K Diplomatic Nearest Neighbour: giving equal chance to all existing classes. Journal of Artificial Intelligence Research.

234. SINGH, M., PROVAN, G.M. (1996) Efficient learning of selective Bayesian network classifiers. 13th International Conference on Machine Learning, Bari, Italy, str. 453–461.

235. SKALAK, D.B. (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. 11th International Conference on Machine Learning, New Brunswick, NJ, USA, str. 293–301.

236. SKALAK, D.B. (1997) Prototype Selection for Composite Nearest Neighbor Classifiers. PhD thesis, Dept. of Computer Science, University of Massachusetts, Amherst, MA, USA.

237. SKUBALSKA-RAFAJŁOWICZ, E., KRZYŻAK, A. (1995) Data sorting along a space filling curve for fast pattern recognition. 2nd International Symposium on Methods and Models in Automation and Robotics, Międzyzdroje, Poland, Vol. 1, str. 339–344.

238. SKUBALSKA-RAFAJŁOWICZ, E., KRZYŻAK, A. (1996) Fast k-NN classification rule using metric on space-filling curves. 13th International Conference on Pattern Recognition, Vienna, Austria, str. 121–125.

239. SNEATH, P.H.A., SOKAL, R.R. (1973) Numerical Taxonomy. W.H.Freeman & Co, San Francisco, CA, USA.

240. SOLLICH, P., KROGH, A. (1996) Learning with ensembles. How overfitting can be useful. W: D.S. Touretzky, M.C. Mozer & M.E. Hesselmo (eds.), Advances in Neural Information Processing Systems, Cambridge, MA, USA, MIT Press, Vol. 8, str. 190–196.

241. SOMOL, P., PUDIL, P., FERRI, F.J., KITTLER, J. (2000) Fast branch & bound algorithm in feature selection. Invited paper for the 4th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL, USA, IIIS, str. 646–651.

242. SPECHT, D.F. (1990) Probabilistic neural networks. Neural Networks, Vol. 3, No. 1, str. 109–118.

243. SPECHT, D.F. (1992) Enhancements to probabilistic neural networks. International Joint Conference on Neural Networks (IJCNN), Vol. 1, str. 761–768.

244. STANFILL, C., WALTZ, D. (1986) Toward memory-based reasoning. Communication of the ACM, Vol. 29, No. 12, str. 1213–1228.

245. STRZECHA, K. (2001) Image segmentation algorithm based on statistical pattern recognition methods. 6th International IEEE Conference on Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv–Slavske, Ukraine, str. 200–201.

246. SWONGER, C.W. (1972) Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. W: S. Watanabe (ed.), Frontiers of Pattern Recognition, Academic Press, New York City, NY, USA, str. 511–519.

247. TADEUSIEWICZ, R. (1993) Sieci neuronowe. Akademicka Oficyna Wydawnicza, Warszawa.

248. TADEUSIEWICZ, R., FLASIŃSKI, M. (1991) Rozpoznawanie obrazów. PWN, Warszawa.

249. TAN, T.-T., DAVIS, L., THURIMELLA, R. (1999) One-dimensional index for nearest neighbor search. European Workshop on Content-Based Multimedia Indexing, Toulouse, France.

250. TOMEK, I. (1976a) A generalization of the k-NN rule. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, No. 2, str. 121–126.

251. TOMEK, I. (1976b) An experiment with the edited nearest-neighbour rule. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-6, No. 6, str. 448–452.

252. TOMEK, I. (1976c) Two modifications of CNN. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-6, No. 11, str. 769–772.

253. TOUSSAINT, G.T. (1994) A counterexample to Tomek's consistency theorem for a condensed nearest neighbor decision rule. Pattern Recognition Letters, Vol. 15, No. 8, str. 797–801.

254. TRAINA JR., C., TRAINA, A.J.M., SEEGER, B., FALOUTSOS, CH. (2000) Slim-trees: High performance metric trees minimizing overlap between nodes. International Conference on Extending Database Technology, Konstanz, Germany, str. 51–65.

255. TRAINA JR., C., TRAINA, A.J.M., WU, L., FALOUTSOS, CH. (2000) Fast feature selection using the fractal dimension. 15th Brazilian Symposium on Databases (SBBD), João Pessoa, Brazil.

256. TUMER, K., GHOSH, J. (1996) Error correlation and error reduction in ensemble classifiers. Connection Science, Vol. 8, No. 3–4, str. 385–404.

257. UHLMANN, J.K. (1991) Satisfying general proximity/similarity queries with metric trees. Information Processing Letters, Vol. 40, Nr. 4, str. 175–179.

258. VAFAIE, H., DE JONG, K. (1992) Genetic algorithms as a tool for feature selection in machine learning. 4th International Conference on Tools with Artificial Intelligence, Arlington, VA, USA, str. 200–204.

259. WEBB, G., PAZZANI, M. (1998) Adjusted probability naive Bayesian induction. 11th Australian Joint Conference on Artificial Intelligence. Brisbane, QLD, Australia, str. 285–295.

260. WEBER, R., SCHEK, H.-J., BLOTT, S. (1998) A quantative analysis and performance study for similarity search methods in high-dimensional spaces. 24th International Conference on Very Large Data Bases (VLDB), New York City, NY, USA, str. 194–205.

261. WETTSCHERECK, D., DIETTERICH, T.G. (1994) Locally adaptive nearest neighbor algorithms. Advances in Neural Information Processing Systems, Vol. 6, str. 184–191.

262. WETTSCHERECK, D., DIETTERICH, T.G. (1995) An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. Machine Learning, Vol. 19, No. 1, str. 5–27.

263. WHITE, D.A., JAIN, R. (1996) Similarity indexing with the SS-tree. 12th International Conference on Data Engineering (ICDE), New Orleans, LA, USA, str. 194–205.

264. WIDROW, B., HOFF JR., M.E. (1960) Adaptive switching circuits. 1960 IRE WESCON Convention Record, str. 96–104. W: Anderson & Rosenfeld, eds. (1988), Neurocomputing — Foundations of Research.

265. WILSON, D.L. (1972) Asymptotic properties of nearest neighbour rules using edited data. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-2, No. 3, str. 408–421.

266. WILSON, D.R. (1997) Advances in Instance-Based Learning Algorithms. PhD thesis, Computer Science Dept., Brigham Young University, Provo, UT, USA.

267. WILSON, D.R., MARTINEZ, T.R. (1997a) Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, Vol. 6, No. 1, str. 1–34.

268. WILSON, D.R., MARTINEZ, T.R. (1997b) Instance pruning techniques. 14th International Conference on Machine Learning, Nashville, TN, USA, str. 403–411.

269. WILSON, D.R., MARTINEZ, T.R. (2000) Reduction techniques for instance-based learning algorithms. Machine Learning, Vol. 38, No. 3, str. 257–286.

270. WINDEATT, T., GHADERI, R. (2000) Multi-class learning and error-correcting code sensitivity. Electronic Letters, Vol. 36, No. 19, str. 1630–1632.

271. WNEK, J., MICHALSKI, R.S. (1994) Discovering representation space transformations for learning concept descriptions combining DNF and M-of-N rules. Working Notes of the ML'94 Workshop on Constructive Induction and Change of Representation, New Brunswick, NJ, USA, str. 61–68.

272. WOLPERT, D.H. (1992) Stacked generalization. Neural Networks, Vol. 5, No. 2, str. 241–259.

273. WOLPERT, D.H. (1993) On overfitting avoidance as bias. Technical Report SFI TR 92-03-5001, The Santa Fe Institute, Santa Fe, NM, USA.

274. WOODS, K., KEGELMEYER, W.P., BOWYER, K. (1997) Combination of multiple classifiers using local accuracy estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 4, str. 405–410.

275. WU, Y., IANAKIEV, K.G., GOVINDARAJU, V. (2002) Improved k-nearest neighbor classification. Pattern Recognition, Vol. 35, str. 2311–2318.

276. YANG, J., HONAVAR, V. (1997) Feature subset selection using a genetic algorithm. 2nd International Conference on Genetic Programming, str. 380–385.

277. YAO, A.C., YAO, F.F. (1985) A general approach to d-dimensional geometric queries. 17th Annual ACM Symposium on Theory of Computing, Providence, RI, USA, str. 163–168.

278. YAO, X., LIU, Y. (1998) Making use of population information in evolutionary artificial neural networks. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-28, No. 3, str. 417–425.

279. YIANILOS, P.N. (1993) Data structures and algorithms for nearest neighbor search in general metric spaces. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA), Austin, TX, USA, str. 311–321.

280. YU, B., YUAN, B. (1993) A more efficient branch and bound algorithm for feature selection. Pattern Recognition, Vol. 26, No. 6, str. 883–889.

281. XIE, Z., WYNNE, H., LIU, Z., LEE, M.-L. (2002) SNNB: A selective neighborhood based naïve Bayes for lazy learning. Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference (PAKDD2002), Taipei, Taiwan, str. 104–114.

282. XU, L., KRZYŻAK, A., SUEN, C.Y. (1992) Methods for combining multiple classifiers and their application to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-22, No. 3, str. 418–435.

283. ZAVREL, J. (1997) An empirical re-examination of weighted voting for k-NN. 7th Belgian-Dutch Conference on Machine Learning (BENELEARN-97), Tilburg, Holland.

284. ZENOBI, G., CUNNINGHAM, P. (2001) Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. 12th European Conference on Machine Learning (ECML), Freiburg, Germany, str. 567–587.

285. ZENOBI, G., CUNNINGHAM, P. (2002) An approach to aggregating ensembles of lazy learners that supports explanation. 6th European Conference on Case-Based Reasoning, Aberdeen, Scotland, str. 436–447.

286. ZHANG, B.-T. (1994) Accelerated learning by active example selection. International Journal of Neural Systems, Vol. 5, No. 1, str. 67–75.

287. ZHANG, J. (1992) Selecting typical instances in instance-based learning. 9th International Conference on Machine Learning, Aberdeen, Scotland, str. 470–479.

288. ZHANG, J., YIM, Y.-S., YANG, J. (1997) Intelligent selection of instances for prediction functions in lazy learning systems. Artificial Intelligence Review, Vol. 11, No. 1–5, str. 175–191.

289. ZHENG, Z. (1998) Naive Bayesian classifier committees. 10th European Conference on Machine Learning, Chemnitz, Germany, str. 196–207.