

Uniwersytet Mikołaja Kopernika
Wydział Fizyki Astronomii i Informatyki Stosowanej
Katedra Informatyki Stosowanej



ROZPRAWA DOKTORSKA

Tomasz Maszczyk

Uniwersalne maszyny uczące

Promotor:
prof. Włodzisław Duch

Toruń, 2012

Spis treści

1	Wstęp	3
2	Cel i zakres pracy	4
3	Transformacja danych wejściowych	5
3.1	Wstępne przetwarzanie danych	5
3.1.1	Normalizacja	6
3.1.2	Standaryzacja	6
3.1.3	Dyskretyzacja cech ciągłych	7
3.1.4	Wartości odstające	7
3.1.5	Wartości brakujące	7
3.2	Selekcja cech	8
4	Metody redukcji wymiarowości i wizualizacji danych	11
4.1	Analiza składników głównych	11
4.2	Skalowanie wielowymiarowe	12
4.3	Analiza dyskryminacyjna Fishera	12
4.4	Wizualizacja triangularyzacyjna	13
4.5	Mapowanie SVMowe	14
5	Techniki klasyfikacji	22
5.1	Klasyfikacja danych	22
5.1.1	Eliminatory	25
5.2	Naiwny klasyfikator Bayesa	25
5.3	Metoda k -najbliższych sąsiadów	26
5.4	Drzewa decyzyjne	26
5.4.1	Porównanie użycia różnych entropii w drzewach	27
5.5	Liniowa dyskryminacja	31
5.6	Sztuczne sieci neuronowe	32
5.7	Maszyny wektorów wspierających	33
5.7.1	Standardowe sformułowanie SVM	33
5.7.2	Kernelowe przestrzenie cech	36
5.8	Maszyny cech wspierających	36
5.9	Maszyny prawie przypadkowych projekcji	40
5.10	Maszyny prawie przypadkowych projekcji z maksymalizacją marginesu	45
5.11	Wygląd granic decyzji dla różnych klasyfikatorów	48

6	Konstrukcja nowych, użytecznych cech	58
6.1	Neurokognitywne inspiracje	58
6.2	Wydobywanie użytecznych cech	59
6.2.1	Wydobywanie informacji z pojedynczych cech	60
6.2.2	Cechy binarne	60
6.2.3	Cechy z wartościami rzeczywistymi	61
6.2.4	Projekcje liniowe	61
6.2.5	Cechy kernelowe	62
6.2.6	Inne nieliniowe mapowania	63
6.2.7	Podsumowanie rodzajów nowych cech	63
6.3	Lokalnie optymalizowane kernele	64
6.4	Uczenie oparte na transformacjach	67
6.4.1	Przededefiniowanie celu uczenia	69
6.4.2	Transfer wiedzy	70
6.5	Uniwersalne maszyny uczące	71
6.6	Wyniki uzyskane przez algorytm ULM	74
7	Podsumowanie	76
A	Charakterystyka zbiorów danych użytych w testach	80
	Bibliografia	82

Rozdział 1

Wstęp

Problem klasyfikacji jest jednym z ważniejszych zagadnień związanych z eksploracją danych znajdującą szerokie zastosowanie w przemyśle, medycynie, nauce i biznesie, dlatego rozwój skutecznych metod inteligencji obliczeniowej jest tak bardzo istotny. Obecnie istnieje wiele algorytmów klasyfikacyjnych, charakteryzujących się różnymi strategiami poszukiwania rozwiązań i różnymi typami możliwych do reprezentacji odwzorowań. Jednak powszechnie używane metody klasyfikacji, pomimo niewątpliwego rozwoju tej dziedziny w ostatnich dziesięcioleciach, ciągle nie są w stanie rozwiązać w możliwie najprostszy sposób wielu stawianych przed nimi problemów. Tworzą one często modele bardzo rozbudowane, które nie zapewniają dostatecznej generalizacji. Utrudnia to także interpretację otrzymanego rozwiązania, a brak możliwości logicznego uwarunkowania otrzymanego rezultatu predykcji jest rzeczą niezwykle istotną w niektórych zastosowaniach (np. w medycynie czy przemyśle).

Inną przyczyną, która znacząco utrudnia możliwość praktycznego wykorzystania niektórych całkiem dobrze działających algorytmów jest fakt, że stanowią one tzw. czarne skrzynki. Tego typu ograniczenia występują zarówno w przypadku sztucznych sieci neuronowych, jak i w przypadku modeli wywodzących się z teorii statystycznych jak np. maszyna wektorów wspierających (*Support Vector Machines, SVM*). Powoduje to, że w wielu przypadkach bardziej powszechne są wszelkiego rodzaju systemy regułowe lub bazujące na prototypach, które umożliwiają zrozumienie procesu podejmowania przez system decyzji.

Zaproponowana w niniejszej rozprawie nowa klasa metod pozwala ominąć te niedogodności. Przykładowo, jawnie określona przestrzeń cech, a następnie odpowiedni prosty klasyfikator pozwalają nie tylko otrzymać wyniki na co najmniej takim poziomie jak wymienione wyżej algorytmy, ale dodatkowo jasno określają w jaki sposób nowe dane zostały zaklasyfikowane.

Budowa dobrego, ale skomplikowanego modelu dla złożonych problemów nie jest najlepszym rozwiązaniem. Proponowane w rozprawie podejście skupia się na poszukiwaniu odpowiednich transformacji przestrzeni wejściowej prowadzących do utworzenia cech charakteryzujących się lepszymi “właściwościami separującymi”. Dodatkowo do tworzenia nowych cech wykorzystywane są informacje pochodzące z innych modeli (np. reguły z drzewa decyzji), które także czynią zadanie znacznie prostszym do rozwiązania. Budowanie modelu dyskryminacyjnego w otrzymanej przestrzeni staje się przez to łatwiejsze.

Rozdział 2

Cel i zakres pracy

Cel:

Celem pracy doktorskiej jest opracowanie i implementacja nowej klasy metod wykorzystujących mechanizmy inteligencji obliczeniowej, opartych na generacji nowych cech, które pozwalają na odkrywanie interesującej wiedzy w danych, transfer wiedzy pomiędzy różnymi modelami danych, a także poszerzanie możliwości znanych metod analizy danych przez wprowadzanie cech nowego typu. Tak więc uwaga autora skupia się bardziej na rozwoju i budowie algorytmów tworzenia nowych cech, niż jak to ma w głównej mierze miejsce w obecnych rozwiązaniach - na optymalizacji parametrów określonego klasyfikatora czy aproksymatora.

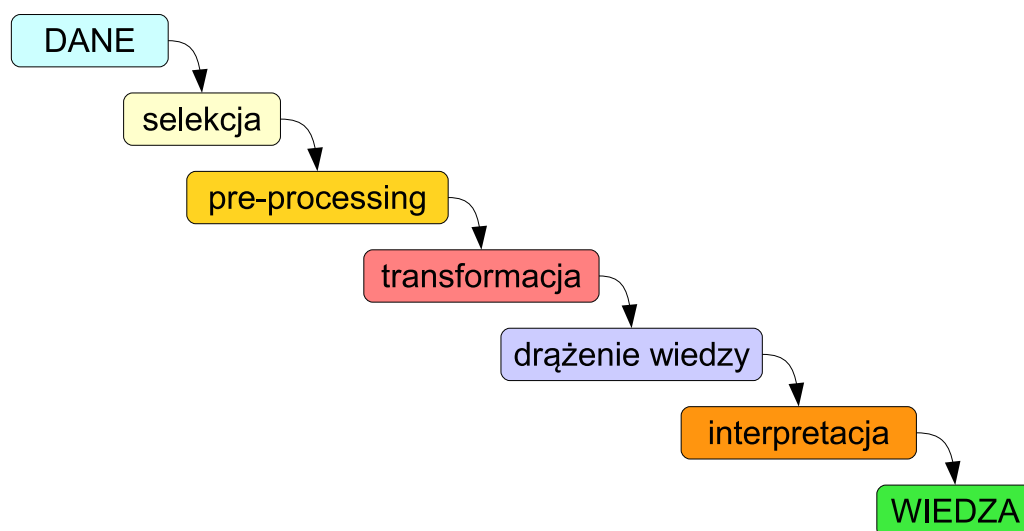
Zakres:

Rozdział 3 niniejszej rozprawy opisuje podstawowe metody wstępnej obróbki danych wejściowych. W kolejnym rozdziale przedstawiono metody służące do redukcji wymiarowości i wizualizacji danych wielowymiarowych (wraz z opisem dwóch autorskich rozwiązań). Rozdział 5 definiuje pojęcie klasyfikacji oraz opisuje różne powszechnie znane metody. Ponadto omówiono w nim (przeanalizowany wcześniej przez autora rozprawy [62]) wpływ zmiany entropii na działanie algorytmu drzew decyzyjnych, a także zaprezentowano kilka nowych metod (maszyny cech wspierających, maszyny prawie przypadkowych projekcji, maszyny prawie przypadkowych projekcji z maksymalizacją marginesu) [65, 15, 60], które mogą być wykorzystane zarówno do klasyfikacji danych, jak i do tworzenia nowych cech. Dodatkowo zilustrowano jak wyglądają granice decyzji dla różnych klasyfikatorów. W rozdziale 6 opisano metody tworzenia nowych, użytecznych cech oraz przedstawiono system integrujący część z zaprezentowanych w tej pracy algorytmów w jedną spójną całość, nazwany Uniwersalnymi Maszynami Uczącymi (*Universal Learning Machines, ULM*) [30], a jego użyteczność została przetestowana i porównana z innymi popularnymi metodami uczenia maszynowego. Ostatni rozdział to próba podsumowania niniejszej pracy doktorskiej.

Rozdział 3

Transformacja danych wejściowych

Proces odkrywania wiedzy w danych składa się z kilku etapów, które zostały pokazane na poniższym schemacie: [34]



Zatem, zanim możliwa będzie jakakolwiek predykcja w oparciu o zgromadzone dane, trzeba je najpierw odpowiednio przygotować. Zajmuje się tym Eksploracyjna Analiza Danych (*Exploratory Data Analysis, EDA*) [89] będąca działem statystyki zmierzającym do podsumowania informacji zawartej w danych, wykonania wstępnych transformacji, redukcji wymiarowości czy wizualizacji danych.

W tym i w następnym rozdziale opisane zostaną procesy wstępnej transformacji danych i redukcji wymiarowości.

3.1 Wstępne przetwarzanie danych

Oryginalne dane przeważnie nie znajdują się w optymalnej postaci dla różnych metod klasyfikacyjnych. Może się okazać, że zastosowanie niektórych transformacji jest konieczne w celu utworzenia bardziej użytecznych cech dla używanych metod. Niestety transformacje, które są optymalne dla jednej metody, mogą się okazać nie najlepszym rozwiązaniem dla innej. Proces wstępnej obróbki danych traktowany jest czasem jako mało znaczący i bardzo często jest pomijany w literaturze. Jednakże w

realnych zastosowaniach okazuje się on być niezwykle istotny, a mianowicie często więcej zależy od wstępnej obróbki danych wejściowych niż od użytej metody klasyfikacji. Poniżej przedstawiono kilka najważniejszych typów transformacji danych, które mogą poprawić rezultaty. Wybór technik i użycie ich w praktycznych aplikacjach zależy od typu danych, ich rozmiaru, i ogólnej charakterystyki postawionego zadania. Wielu specjalistów z dziedziny uczenia maszynowego przyznaje, że jedną z kluczowych ról w procesie klasyfikacji odgrywa proces wstępnej obróbki i transformacji oryginalnych danych. Sam proces wstępnego przygotowania danych ma istotny wpływ zarówno na szybkość uczenia modelu jak i na jego generalizację.

Bardzo często zdarza się, że w oryginalnych danych cechy znacznie się od siebie różnią zakresem przyjmowanych wartości (zależnie od rodzaju wybranej skali pomiarowej). Część metod działa w taki sposób, że za bardziej istotną uznaje tę cechę, której wartości są większe, zamiast wziąć tę, która dostarcza najwięcej informacji. Aby ujednolicić wpływ poszczególnych cech niezależnie od wybranej skali pomiarowej, można zastosować kilka transformacji. Najbardziej znanymi są normalizacja i standaryzacja.

3.1.1 Normalizacja

W wyniku tej transformacji otrzymuje się cechy, których wartości należą do przedziału $[0, 1]$.

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \quad (3.1)$$

gdzie:

$x_{\min}$ – minimalna wartość cechy i w zbiorze treningowym,

$x_{\max}$ – maksymalna wartość cechy i w zbiorze treningowym.

Zarówno dla zbioru treningowego jak i testowego normalizacja przebiega w ten sam sposób, tzn dla obu zbiorów zostają użyte $x_{\min}$ i $x_{\max}$ wyznaczone dla zbioru treningowego.

Ten typ przekształcenia może być czasem niebezpieczny, gdyż nie bierze on pod uwagę rozkładu wartości danej cechy. Może to spowodować, że w przypadku wystąpienia wartości odstających (znacznie różnych od wartości typowych) w wyniku normalizacji większość wartości danej cechy znajdzie się w bardzo wąskim przedziale.

3.1.2 Standaryzacja

Wykorzystuje ona rozkład wartości w poszczególnych cechach i przeprowadzana jest zgodnie ze wzorem:

$$x'_i = \frac{x_i - \bar{x}}{\sigma_x} \quad (3.2)$$

gdzie średnia wartość cechy x_i :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (3.3)$$

a odchylenie standardowe cechy x_i :

$$\sigma_x = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (3.4)$$

Transformacja taka powoduje, że otrzymane wartości dla każdej z cech mają rozkład normalny o średniej równej zero i odchyleniu standardowym równym jeden.

3.1.3 Dyskretyzacja cech ciągłych

Ten typ transformacji używany jest do konwersji danych ciągłych na dane dyskretne. Dzieje się to przez podział cechy ciągłej na rozłączne i zarazem pokrywające jej zakres przedziały. Proces ten odbywa się w taki sposób, aby przynależność do poszczególnych odcinków zawierała w sobie maksymalną ilość informacji o klasie. Dyskretyzacja wpływa m.in. na zwiększenie efektywności obliczeniowej, zwiększa czytelność, prostotę i dokładność hipotez. Jednakże osiągnięcie jak największej korzyści z tego typu przekształcenia danych wymaga aby ilość wartości dyskretnych była jak najmniejsza.

Występuje kilka rodzajów dyskretyzacji, i tak jeśli uwzględniać to, czy brana jest pod uwagę etykieta klasy, do której należy dany wektor, można wyróżnić dyskretyzację z nadzorem i bez nadzoru. Gdy do transformacji uwzględniać się będzie rozkład wartości atrybutów i klas, wówczas będzie to dyskretyzacja zaawansowana, w przeciwnym razie jest to tzw. dyskretyzacja prymitywna. Ostatnim typem podziału dyskretyzacji jest zakres jej działania, a więc tutaj występują typ lokalny i globalny. Jak zatem widać, przekształcenie to niesie za sobą wiele korzyści, a jednocześnie pozwala na użycie niektórych algorytmów, które źle radzą sobie z cechami ciągłymi.

3.1.4 Wartości odstające

Definiuje się je mówiąc, że są to takie przypadki, których wartości znacznie odbiegają od przeciętnych występujących w danym zbiorze. Dostyć często występują one w dużych zbiorach danych. Część przypadków może znaleźć się daleko od wartości średniej lub też mogą one powstać na skutek błędnych pomiarów, zapisów albo też mogą się zdarzyć przypadki pochodzące z innej populacji (danych o innym rozkładzie).

Wiele algorytmów znacznie gorzej radzi sobie z danymi, wśród których występują wartości odstające. Aby wyeliminować ich niekorzystny wpływ na funkcjonowanie budowanego modelu stosuje się różne podejścia. Pierwszym z nich jest budowa dwóch osobnych modeli: jeden dla danych normalnych oraz drugi dla danych odstających i w zależności, z jakim przypadkiem testowym ma się do czynienia, zastosowanie tego właściwego. Innym podejściem (najczęściej używanym) jest usunięcie wektorów odstających, jednakże w tym przypadku należy bardzo uważać, aby nie usunąć wektorów zawierających jakieś istotne informacje. Rozróżnia się także kilka metod wykrywania wartości odstających. Jednym z nich jest podejście bazujące na pomiarze odległości do k -najbliższych sąsiadów i użycie ich do określenia czy dana obserwacja jest normalna. Inne często używane metody bazują na wyliczonych wartościach średniej i odchylenia standardowego. Zaliczyć do nich można kryterium Chauveneta [4].

3.1.5 Wartości brakujące

W przypadku wielu zbiorów danych pojawia się problem wartości brakujących, tzn. występują takie wektory, dla których wartość jednej lub wielu cech nie jest określona. Część algorytmów potrafi poradzić sobie z takimi niekompletnymi danymi (np. drzewa decyzyjne), jednakże większość wymaga aby wszystkie cechy były określone.

Najprostszą metodą pozbycia się niedogodności występowania wartości brakujących jest usunięcie wszystkich wektorów zawierających braki. Jest to możliwe jedynie w przypadku, gdy ilość takich wektorów nie jest zbyt duża w porównaniu do całkowitej liczby wektorów występujących w zbiorze treningowym. Inną często stosowaną metodą jest uzupełnianie braków średnią wartością atrybutu wyliczoną dla wszystkich wektorów, które miały w danej ceście wartość określoną (z wykorzystaniem informacji o klasach - wówczas dla każdej z klas średnia jest liczona oddzielnie, lub bez wykorzystania informacji o klasach - wyliczana jest jedna średnia dla każdej z cech i tą wartością uzupełnia się braki). Niestety takie rozwiązanie nie daje zwykle dobrych rezultatów. Znacznie lepsze wyniki dają metody oparte na wykorzystaniu informacji znajdujących się w pozostałych cechach. Najbardziej znane z nich to:

1. Metoda k -najbliższych sąsiadów – opiera się ona na uzupełnianiu wartości brakujących, wartościami średnimi wyliczonymi dla k najbliższych wektorów. To które wektory zostaną uznane za najbliższe zależy od wybranej miary odległości.
2. Maksymalizacja wartości oczekiwanej – metoda ta oparta jest na założeniu, że dla wartości obserwowanych $\mathbf{X}_{obs}$ i wartości brakujących występujących w danych $\mathbf{X}_{brak}$ prawdopodobieństwo łączne wynosi $p(\mathbf{X}_{obs}, \mathbf{X}_{brak}|\theta)$, gdzie θ to parametry modelu. A zatem prawdopodobieństwo brzegowe dla wartości obserwowanych wynosi:

$$p(\mathbf{X}_{obs}|\theta) = \sum_{\mathbf{X}_{brak}} p(\mathbf{X}_{obs}, \mathbf{X}_{brak}|\theta) \quad (3.5)$$

Jest to algorytm iteracyjny i składa się on z dwóch etapów:

- wyznaczenia prawdopodobieństwa dla wartości brakujących,

$$p_n(\mathbf{X}_{brak}) = p(\mathbf{X}_{brak}|\mathbf{X}_{obs}, \theta^{n-1}) \quad (3.6)$$

- maksymalizacji prawdopodobieństwa (zakładając brak wartości brakujących), w wyniku której otrzymywane są nowe parametry modelu estymacji.

$$\theta^n = \max_{\theta} E(p(\mathbf{X}_{brak}, \mathbf{X}_{obs}|\theta)) \quad (3.7)$$

Kroki te powtarza się do momentu, aż zmiana parametrów modelu estymacji będzie pomijalnie mała.

3. Wielokrotne przypisanie – technika ta jest nieco podobna do poprzedniej, aczkolwiek dodatkowo wprowadza się element niepewności. Każda z wartości brakujących może przyjąć (otrzymane z przewidywanego rozkładu) $m > 1$ wartości. A zatem wymagane jest utworzenie m zbiorów z jednakowymi wartościami obserwowanymi, oraz wartościami brakującymi wypełnionymi w każdym ze zbiorów inną wartością. Różnice te wynikają z niepewności z jaką może być przewidziana każda z wartości brakujących [78].

3.2 Selekcja cech

Selekcja cech jest bardzo ważnym etapem w trakcie przygotowywania danych do późniejszej klasyfikacji lub regresji. Teoretycznie duża liczba cech powinna dać nam znaczne możliwości przy predykcji, jednakże jest wiele przesłanek ku temu, aby

przeprowadzić redukcję dostępnych cech do niezbędnego minimum. Można zaliczyć tutaj m.in.:

- większa liczba cech to także większa złożoność obliczeniowa,
- cechy mogą być wzajemnie skorelowane,
- przekleństwo wymiarowości [5].

Aby wybrać pewną liczbę n cech, które najlepiej różnicują klasy (w przypadku zadania klasyfikacyjnego), należy zdefiniować funkcje, które będą mierzyć dobroć klasyfikatora, a następnie wyznaczyć wartość owych funkcji dla każdego możliwego podzbioru n cech. W tym miejscu pojawia się problem, gdyż chcąc wyznaczyć optymalny podzbiór n spośród wszystkich m cech, należy sprawdzić:

$$\binom{m}{n} = \frac{m!}{n!(m-n)!} \quad (3.8)$$

różnych podzbiorów. Dodatkowo należy mieć na uwadze, że tak naprawdę nie jest znana liczba elementów n , które będą dawać najlepszy wynik. Tak więc liczba wszystkich podzbiorów jakie należy sprawdzić wynosi:

$$\sum_{i=1}^m \binom{m}{i} \quad (3.9)$$

Aby ominąć tak dużą złożoność obliczeniową stosuje się różne metody. Jedną z nich jest metoda, która polega na indywidualnym traktowaniu każdej cechy, czyli ranking przydatności poszczególnych cech. Dla każdego z atrybutów oblicza się wartość funkcji dyskryminacji, a następnie wybiera się takie cechy, dla których funkcja przyjmuje największe wartości. Takie podejście nie jest pozbawione niestety wad. Największa z nich to fakt, iż w tym przypadku nie uwzględnia się sytuacji, w której sama cecha ma słaby wpływ na końcowy wynik, ale w połączeniu z innymi daje bardzo dobre rezultaty.

Algorytmy metod selekcji cech można podzielić na różne sposoby [14], jednak biorąc pod uwagę jako kryterium podziału strategie oceniania przydatności cechy, selekcję cech można podzielić na:

- metodę powłoki – jest to metoda dwustopniowa, w której znaczącą rolę przy wyszukiwaniu istotnych cech odgrywa zarówno algorytm selekcji cech jak i algorytm uczący [52],
- metodę filtrów – oceniają one cechy oraz ich właściwości wykorzystując dostępne dane [57, 20].

Do oceny pojedynczych cech i uporządkowania ich według przewidywanej przydatności (zazwyczaj od cechy najbardziej do najmniej przydatnej) służą metody rankingowe. Można spotkać wiele funkcji kryterialnych, zarówno opartych o teorię informacji jak i bazujących na miarach statystycznych. Obecnie najbardziej popularne metody rankingowe to:

- suma wariancji

$$\sum_{i=1}^{n_l} c_i^2 - \frac{l}{n_l} \left(\sum_{i=1}^{n_l} c_j \right)^2 + \sum_{i=1}^{n_r} c_i^2 - \frac{l}{n_r} \left(\sum_{i=1}^{n_r} c_j \right)^2 \quad (3.10)$$

- indeks gini

$$\frac{n_l}{n} \left(1 - \sum_{i=1}^k \left(\frac{l_i}{n_l} \right)^2 \right) + \frac{n_r}{n} \left(1 - \sum_{i=1}^k \left(\frac{r_i}{n_r} \right)^2 \right) \quad (3.11)$$

- zysk informacji

$$\sum_{i=1}^k \left(\frac{l_i}{n} \log \frac{l_i}{n_l} + \frac{r_i}{n} \log \frac{r_i}{n_r} \right) - \sum_{i=1}^k \left(\frac{l_i + r_i}{n} \right) \log \left(\frac{l_i + r_i}{n} \right) \quad (3.12)$$

- maksimum mniejszości

$$\max \left(\left(\sum_{i=1}^k l_i - \max(l_i) \right), \left(\sum_{i=1}^k r_i - \max(r_i) \right) \right) \quad (3.13)$$

- suma mniejszości

$$\left(\sum_{i=1}^k l_i - \max(l_i) \right) + \left(\sum_{i=1}^k r_i - \max(r_i) \right) \quad (3.14)$$

Następnie w takim podejściu dokonuje się wyboru odpowiedniego podzbioru cech. W odróżnieniu do przedstawionych przed chwilą algorytmów rankingowych wyróżnia się jeszcze inne podejście, zwane ogólnie algorytmami przeszukiwania [91, 83]. Należy w nich ustalić takie rzeczy jak punkt startowy w przestrzeni poszukiwań, który wyznaczy kierunek przeszukiwania (selekcja w przód lub selekcja w tył). Metody te charakteryzuje dużo większa dokładność, a otrzymywane rezultaty z reguły przewyższają wyniki metod rankingowych. Niestety wadą ich jest to, że wymagają dużego nakładu obliczeniowego. Można zatem pokusić się o stwierdzenie, że w przypadku dużych zbiorów danych rozsądniejsze wydaje się być użycie metod rankingowych, w przeciwnym wypadku lepiej użyć algorytmów przeszukiwania.

Przy wyborze odpowiedniej metody należy także zwrócić uwagę na stabilność procesu selekcji cech. Definiuje się ją jako podatność algorytmu na różne perturbacje w danych, pochodzące najczęściej od jakiegoś szumu. Obecność szumu jest wszechobecna, dlatego też dobry algorytm selekcji cech powinien być na niego odporny i zapewniać stabilne rezultaty obejmujące tylko istotne cechy. Aby zmierzyć stabilność algorytmu selekcji cech, można wygenerować dwa zbiory danych powstałe w wyniku dodania szumu do jakichś danych oryginalnych, a następnie ocenić podobieństwo list cech otrzymanych w wyniku działania algorytmu na tych zbiorach [51].

Rozdział 4

Metody redukcji wymiarowości i wizualizacji danych

Wiele problemów w uczeniu maszynowym nie może zostać rozwiązanych bez redukcji wymiarowości lub analizy struktury danych, która pomaga w wyborze odpowiedniego algorytmu klasyfikującego. Warto także zaznaczyć, że niektóre kosztowne obliczeniowo algorytmy wymagają redukcji wymiarowości po to, aby poradzić sobie z dużymi zbiorami danych. Dlatego wybór modelu powinien nie tylko skupiać się na poszukiwaniu właściwego klasyfikatora (czy aproksymatora), gdyż dobór wspomagających transformacji jest także bardzo istotny.

Zrozumienie struktury danych możliwe jest poprzez wizualizację i redukcję wymiarowości. Tak więc wiedza uzyskana dzięki eksploracji danych, w połączeniu z odpowiednimi transformacjami, umożliwia zbudowanie najprostszych modeli danych. Jest to jedno z najbardziej skutecznych podejść, prowadzące do poszukiwania najlepszych modeli wśród danych, które zostały poddane różnym transformacjom. Umożliwia to rozwiązanie problemów o skomplikowanej, logicznej strukturze, które są bardzo trudne dla tradycyjnych algorytmów uczenia maszynowego.

W następnej części omówionych zostanie kilka popularnych algorytmów redukcji wymiarowości (używanych także do wizualizacji danych wielowymiarowych): analiza składników głównych (PCA), analiza dyskryminacyjna Fishera (FDA) oraz skalowanie wielowymiarowe (MDS). Następnie przedstawione zostaną dwie autorskie metody: wizualizacja triangularyzacyjna [66] oraz redukcja SVMowa [63].

4.1 Analiza składników głównych

Analiza składników głównych (*Principal Component Analysis, PCA*) to metoda liniowego rzutowania, która znajduje ortogonalne kombinacje cech wejściowych $\mathbf{X} = \{x_1, x_2, \dots, x_N\}$, gdzie każdy nowy kierunek zawiera największą pozostałą wariancję danych. Ponieważ metoda ta nie używa etykiet klas, może być zastosowana do każdego dostępnego zbioru danych. Składniki główne $\mathbf{P}_i$ otrzymane z diagonalizacji macierzy kowariancji, gwarantują minimalną stratę informacji, kiedy pozycja punktów zostanie odtworzona z ich nisko-wymiarowych projekcji. Biorąc 1, 2 lub 3 składniki główne i rzutując dane na przestrzeń zdefiniowaną przez te składniki $y_{ij} = \mathbf{P}_i \cdot \mathbf{X}_j$, otrzymuje się dla każdego wektora wejściowego jego odwzorowanie $\mathbf{Y}_j = (y_{1j}, y_{2j}, \dots, y_{kj})$ w przestrzeni wyjściowej.

4.2 Skalowanie wielowymiarowe

Skalowanie wielowymiarowe (*Multidimensional Scaling, MDS*) jest nieliniową techniką używaną do wizualizacji sąsiedztwa [13, 7, 55]. Główna idea opiera się na zmniejszeniu wymiarowości danych, przy jednoczesnym zachowaniu oryginalnych odległości między punktami danych tak, jak jest to zdefiniowane w wysokowymiarowej przestrzeni. MDS wymaga jedynie podobieństwa pomiędzy obiektami, a zatem jawna reprezentacja przypadków wejściowych nie jest konieczna. W skalowaniu metrycznym używana jest ilościowa ocena podobieństwa oparta na numerycznych miarach odległości (Euklidesowa, kosinusowa lub inna). W przypadku skalowania niemetrycznego używa się jakościowej informacji o podobieństwie par punktów. Metody MDS różnią się ze względu na ich funkcję kosztu, algorytmy optymalizacji, liczbę używanych macierzy podobieństwa, oraz użycie ważenia cech. Istnieje wiele miar topograficznych wypaczeń otrzymywanych przez redukcję wymiarowości, jednak większość z nich to różne warianty następującej funkcji stresu:

$$S_T(\mathbf{d}) = \sum_{i>j}^n (D_{ij} - d_{ij})^2 \quad (4.1)$$

gdzie:

d_{ij} – odległości w docelowej (nisko-wymiarowej) przestrzeni,

D_{ij} – odległości w przestrzeni wejściowej wyliczone przy użyciu funkcji metrycznych.

Miary te są minimalizowane poprzez modyfikacje pozycji wszystkich punktów docelowych. W k -wymiarowej przestrzeni docelowej jest $kn - 3$ parametrów do minimalizacji [24].

Do wizualizacji używa się zwykle 1, 2 lub 3 wymiarowej przestrzeni docelowej. Suma liczona jest po wszystkich parach wektorów, zatem złożoność algorytmu to $O(n^2d)$. Funkcje kosztu MDS nie są proste do minimalizacji, zawierają bowiem wiele lokalnych minimum, z których każde odzwierciedla inne mapowanie. Wstępna konfiguracja jest albo wybierana losowo, albo oparta na rzucie danych do przestrzeni wyznaczonej przez składniki główne. Orientacja osi w mapowaniu MDS jest umowna, zatem wartości współrzędnych nie mają żadnej prostej interpretacji, gdyż istotne są jedynie odległości między punktami.

4.3 Analiza dyskryminacyjna Fishera

Analiza dyskryminacyjna Fishera (*Fisher Discriminant Analysis, FDA*) jest metodą z nadzorem, zatem używa ona informacji o klasach w celu znalezienia takich rzutów, które separują przypadki z tych klas. Ten popularny algorytm maksymalizuje stosunek między-klasowego rozrzutu do wewnątrz-klasowego rozrzutu, poszukując kierunku $\mathbf{W}$ takiego że:

$$\max_{\mathbf{W}} J_{\mathbf{W}} = \frac{\mathbf{W}^T \mathbf{S}_B \mathbf{W}}{\mathbf{W}^T \mathbf{S}_I \mathbf{W}} \quad (4.2)$$

gdzie rozrzuty macierzy $\mathbf{S}_B$ i $\mathbf{S}_I$ są zdefiniowane przez:

$$\mathbf{S}_B = \sum_{i=1}^C \frac{n_i}{n} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T; \quad \mathbf{S}_I = \sum_{i=1}^C \frac{n_i}{n} \hat{\Sigma}_i \quad (4.3)$$

gdzie $\mathbf{m}_i$ i $\hat{\Sigma}_i$ to średnia i kowariancja macierzy dla każdej klasy, zaś $\mathbf{m}$ jest średnim wektorem z wszystkich przypadków [91].

FDA jest często używane do klasyfikacji rzutując dane na prostą. Do wizualizacji utworzenie drugiego wektora FDA w dwuklasowym problemie nie jest trywialne. Dzieje się tak przez fakt, że rząd $\mathbf{S}_B$ macierzy dla C -klasowego problemu wynosi $C - 1$. Istnieje kilka rozwiązań tego problemu: [11, 91]

- stabilizacja macierzy $\mathbf{S}_I$ przez dodanie małej macierzy perturbacji,
- użycie pseudoinwersji, zastępując $\mathbf{S}_I^{-1}$ przez macierz pseudoinwersji $\mathbf{S}_I^\dagger$,
- użycie stopniowej dekompozycji macierzy.

4.4 Wizualizacja triangularyzacyjna

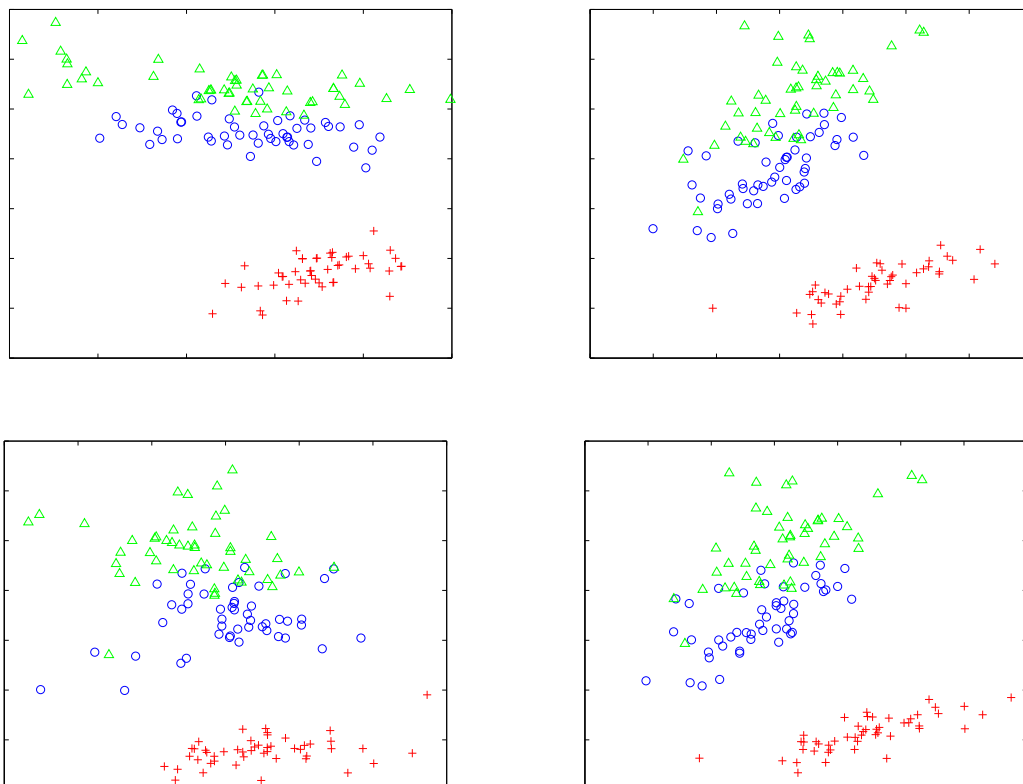
Algorytm wizualizacji triangularyzacyjnej (*Triangular Visualisation*, *TriVis*) tworzy reprezentację oryginalnych danych w dwuwymiarowej przestrzeni, przy jednoczesnym zachowaniu tylu odległości pomiędzy wektorami danych, ile jest tylko możliwe. Odległości pomiędzy dowolnymi trzema wektorami tworzącymi trójkąt mogą zostać zawsze poprawnie odwzorowane. Każdy kolejny wektor jest iteracyjnie dodawany do jednej ze ścian istniejącego trójkąta, tworząc nowy trójkąt, który zachowuje dokładnie dwie oryginalne odległości. Istnieje wiele możliwości dodania takich punktów w relacji do istniejących ścian trójkątów. Aby zachować ogólną strukturę danych, początkowe trzy punkty wybierane są w taki sposób, aby ich wzajemna suma odległości od siebie była największa, zaś w kolejnym kroku dodawany jest taki punkt, który minimalizuje MDSową funkcję stresu $S(\mathbf{d}) = \sum_{i>j}^n (D_{ij} - d_{ij})^2$. Prowadzi to do powstania takiego odwzorowania, które zachowuje dokładnie $2n - 3$ spośród $n(n - 1)/2$ oryginalnych odległości, przy minimalizacji ogólnego stresu.

Schemat 1 Wizualizacja triangularyzacyjna

- 1: Znajdź trzy najdalsze wektory i oznacz je (zachowując oryginalne odległości) jako wierzchołki pierwszego trójkąta.
 - 2: Oznacz segmenty (pary punktów) tworzące boki trójkąta jako “dostępne”.
 - 3: **for** $i = 1$ to $n - 3$ **do**
 - 4: Znajdź taki segment AB, dla którego wektor $\mathbf{X}_i$ dodany jako punkt $C=C(\mathbf{X}_i)$ utworzy trójkąt ABC, przy zachowaniu dwóch oryginalnych odległości $|AC|$ oraz $|BC|$ i dodatkowo spowoduje najmniejszy wzrost stresu $S_i = \sum_{j=1}^m (D_{ij} - d_{ij})^2$.
 - 5: Usuń segment AB z listy “dostępnych” segmentów i dodaj do niej segmenty AC oraz BC.
 - 6: **end for**
-

Dodatkowo algorytm TriVis może zostać użyty do inicjalizacji metody MDS. Zabieg ten powoduje zmniejszenie stresu na samym początku i co za tym idzie, znaczne ograniczenie liczby iteracji.

Aby pokazać jak działa wizualizacja triangularyzacyjna, oraz jaki jest wpływ użycia jej do inicjalizacji MDS, przeprowadzono kilka eksperymentów. Dla każdego zbioru danych dokonano mapowania przy użyciu PCA, TriVis, MDS zainicjowanego w sposób losowy, oraz MDS zainicjalizowanego przy pomocy algorytmu TriVis (rys. 4.1-4.4).



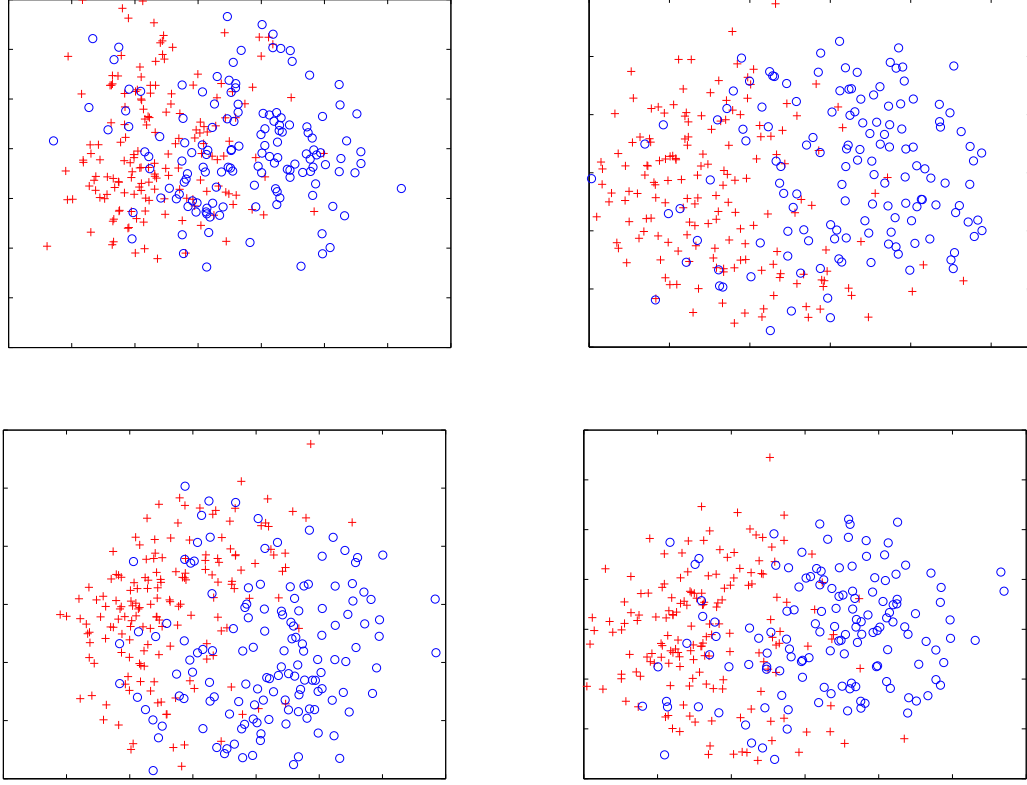
Rysunek 4.1: Zbiór *Iris*; górny wiersz: PCA i TriVis, dolny wiersz: typowy (losowo inicjalizowany) MDS i MDS inicjalizowany przez TriVis.

Wizualizacja zarówno zbioru *Iris* jak i *Heart-c* (opis zbiorów w dodatku A) wygląda raczej podobnie dla wszystkich prezentowanych tutaj technik (zob. rysunki 4.1 i 4.2), a wpływ zmiany rodzaju inicjalizacji MDS w tych przypadkach jest raczej znikomy. Zbiór *Wine*, jak widać na rysunku 4.3, nie mapuje się zbyt dobrze do dwóch wymiarów przy użyciu PCA, za to pozostałe metody radzą sobie całkiem przyzwoicie. Widać także pozytywny wpływ inicjalizacji MDSu przy użyciu algorytmu TriVis, dzięki której otrzymuje się lepszą separację danych, przy jednoczesnym ograniczeniu liczby niezbędnych iteracji. Przypadek *Leukemii* (zob. rysunek 4.4) pokazuje dobre odseparowanie klas za pomocą TriVis, która na tym zbiorze wydaje się prezentować ciekawsze rozwiązanie od pozostałych metod.

Działanie różnych typów inicjalizacji MDS zaprezentowano na rysunku 4.5. Pozytywny wpływ inicjalizacji przy użyciu metody TriVis jest dobrze widoczny, prowadzi ona do szybszej minimalizacji funkcji stresu (mniejsza ogólna liczba iteracji) i powoduje, że jej końcowe wartości są mniejsze. Warto dodać, że ten typ inicjalizacji może uchronić algorytm MDS przed utknięciem w jakichś lokalnych minimach.

4.5 Mapowanie SVMowe

Liniowy SVM (*Support Vector Machines*, *SVM*) to algorytm tworzący hiperpłaszczyznę, która zapewnia duży margines klasyfikacji, używając do tego regularyzacji i programowania kwadratowego. Nieliniowe wersje działają w oparciu



Rysunek 4.2: Zbiór *Heart-c*; górny wiersz: PCA i TriVis, dolny wiersz: typowy MDS i MDS inicjalizowany przez TriVis.

o kernelowy trik [80], który umożliwia dokonanie mapowania wektorów wejściowych do wielowymiarowej przestrzeni cech, w której konstruowana jest najlepiej separująca hiperpłaszczyzna (hiperpłaszczyzna z maksymalnym marginesem). Liniowa funkcja dyskryminacyjna jest określona przez:

$$g_{\mathbf{W}}(\mathbf{X}) = \mathbf{W}^T \cdot \mathbf{X} + w_0 \quad (4.4)$$

Najlepiej dyskryminująca hiperpłaszczyzna powinna maksymalizować odległość pomiędzy hiperpłaszczyzną decyzji zdefiniowaną przez $g_{\mathbf{W}}(\mathbf{X}) = 0$ i wektorami, które są najbliższe niej $\max_{\mathbf{W}} D(\mathbf{W}, \mathbf{X}^{(i)})$. Największy margines klasyfikacji jest osiągany poprzez minimalizację normy $\|\mathbf{W}\|^2$ z ograniczeniami:

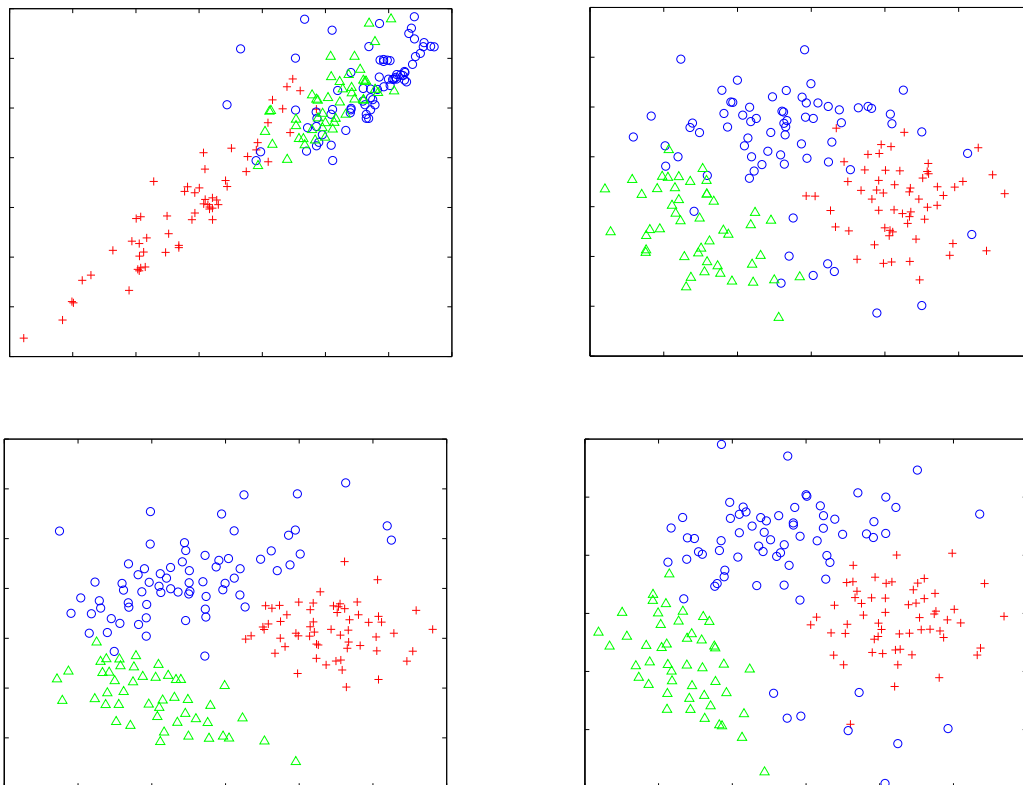
$$\mathbf{Y}^{(i)} g_{\mathbf{W}}(\mathbf{X}^{(i)}) \geq 1 \quad (4.5)$$

dla wszystkich wektorów treningowych $\mathbf{X}^{(i)}$ należących do klasy $\mathbf{Y}^{(i)}$. Wektor $\mathbf{W}$, ortogonalny do hiperpłaszczyzny dyskryminującej, wyznacza kierunek, na który wektory wejściowe są rzutowane i takie mogą być użyte do budowy jedno-wymiarowej projekcji.

To samo może zostać zrealizowane przy użyciu nieliniowego SVMa bazującego na dyskryminacji kernelowej:

$$g_{\mathbf{W}}(\mathbf{X}) = \sum_{i=1}^{N_{sv}} \alpha_i K(\mathbf{X}^{(i)}, \mathbf{X}) + w_0 \quad (4.6)$$

gdzie sumowanie występuje po tych wektorach wsparcia $\mathbf{X}^{(i)}$, które zostały wybrane ze zbioru treningowego.



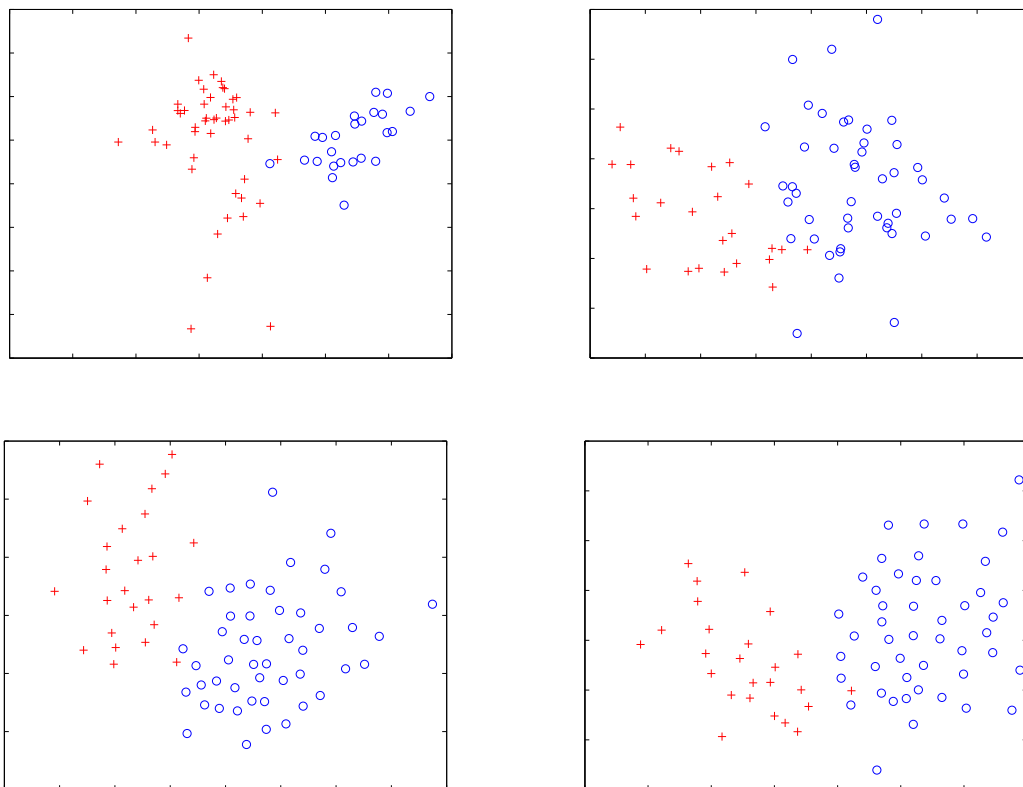
Rysunek 4.3: Zbiór *Wine*; górny wiersz: PCA i TriVis, dolny wiersz: typowy MDS i MDS inicjalizowany przez TriVis.

Wartości $x = g_{\mathbf{W}}(\mathbf{X})$ dla przeciwnych klas mogą zostać wygładzone i przedstawione przy pomocy histogramu, określając prawdopodobieństwa warunkowe $p(\mathbf{X}|C)$ albo prawdopodobieństwa posteryoryczne $p(C|\mathbf{X}) = p(\mathbf{X}|C)p(C)/p(\mathbf{X})$ (patrz rys. 4.8).

Wizualizacja SVM w więcej niż jednym wymiarze wymaga wygenerowania dodatkowych kierunków dyskryminujących. Pierwszy rzut na $\mathbf{W}_1$ powinien dać $g_{\mathbf{W}_1}(\mathbf{X}) < 0$ dla wektorów z pierwszej klasy i $g_{\mathbf{W}_1}(\mathbf{X}) > 0$ dla drugiej. Jest to oczywiście możliwe jedynie dla danych liniowo separowalnych. Kolejne kierunki mogą zostać znalezione w przestrzeni ortogonalnej do pierwszego kierunku, więc proces ten może być powtarzany w celu otrzymania większej liczby wymiarów. Każdy dodatkowy wymiar powinien pomóc zmniejszyć ilość błędów, zaś optymalna wymiarowość zostaje osiągnięta wówczas, gdy nowe wymiary przestaną zmniejszać liczbę błędów występujących w testach krosvalidacji.

Poniżej porównano wizualizację przy użyciu metod MDS, PCA, FDA i SVM dla kilku zbiorów danych. Ponadto zaprezentowano wyniki klasyfikacji owych zbiorów przy użyciu klasyfikatora SVM z kernelem liniowym. Uczyniono to zarówno dla zbiorów w oryginalnej postaci, jak i po ich redukcji do 1 lub 2 wymiarów za pomocą jednej z wymienionych powyżej technik.

Problem wysokowymiarowej parzystości nie jest trywialny i tak jak widać na rysunku 4.6 nie występuje tutaj liniowa separowalność, gdyż jest to problem k -separowalny, czyli w n wymiarach minimalna liczba jednorodnych klastrów po liniowej projekcji wynosi $n + 1$ [40, 26]. PCA i SVM znajduje w tym przypadku bardzo użyteczny pierwszy kierunek rzutowania, podczas gdy drugi rzut nie ma już większego znaczenia. MDS



Rysunek 4.4: Zbiór *Leukemia*; górny wiersz: PCA i TriVis, dolny wiersz: typowy MDS i MDS inicjalizowany przez TriVis.

kompletnie sobie nie radzi z tym zbiorem, a FDA pokazuje niewielkie nakładanie się klastrów w pierwszym wymiarze. Jest to znakomity przykład pokazujący, że wizualizacja lub rzutowanie problemu do mniejszej liczby wymiarów może pomóc uprościć zadanie klasyfikacji. Warto zauważyć, że nawet po odpowiednim mapowaniu liniowe metody nie poradzą sobie z tym problemem, koniecznym wydaje się być tutaj użycie drzewa decyzyjnego lub algorytmu najbliższych sąsiadów.

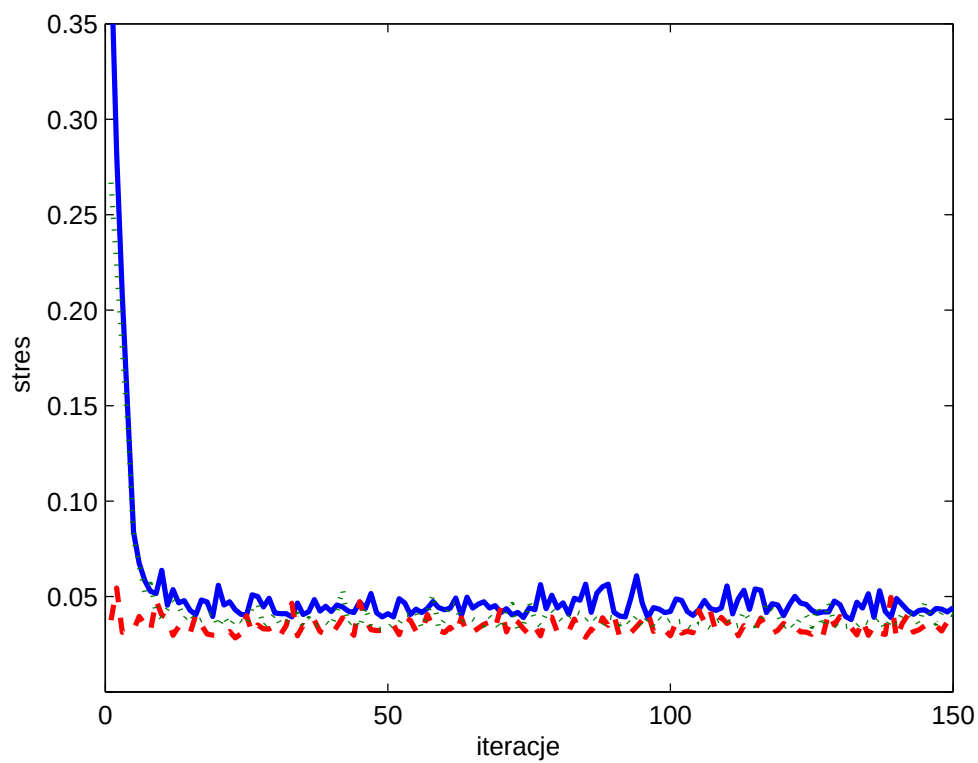
Dla *Heart-c* SVM pokazuje całkiem niezłą separację sporej części danych (rys. 4.7), przy jednoczesnej niewielkiej poprawie po dodaniu drugiego kierunku. Rzutowania MDS i PCA okazują się być mniej użyteczne niż FDA czy SVM.

Wykreślenie prawdopodobieństw warunkowych w zależności od klasy dla *Parity8* i *Heart-c* (opis zbiorów w dodatku A) w pierwszym kierunku uzyskanym przez metodę SVM (rys. 4.8) może pomóc w ocenie nakładania się danych na siebie, a także wydaje się być pomocne przy wyborze rodzaju transformacji, która zostanie użyta przed rzutowaniem (to samo wykreślono także dla zbiorów *Breast-w* i *Leukemia* (rys. 4.11)).

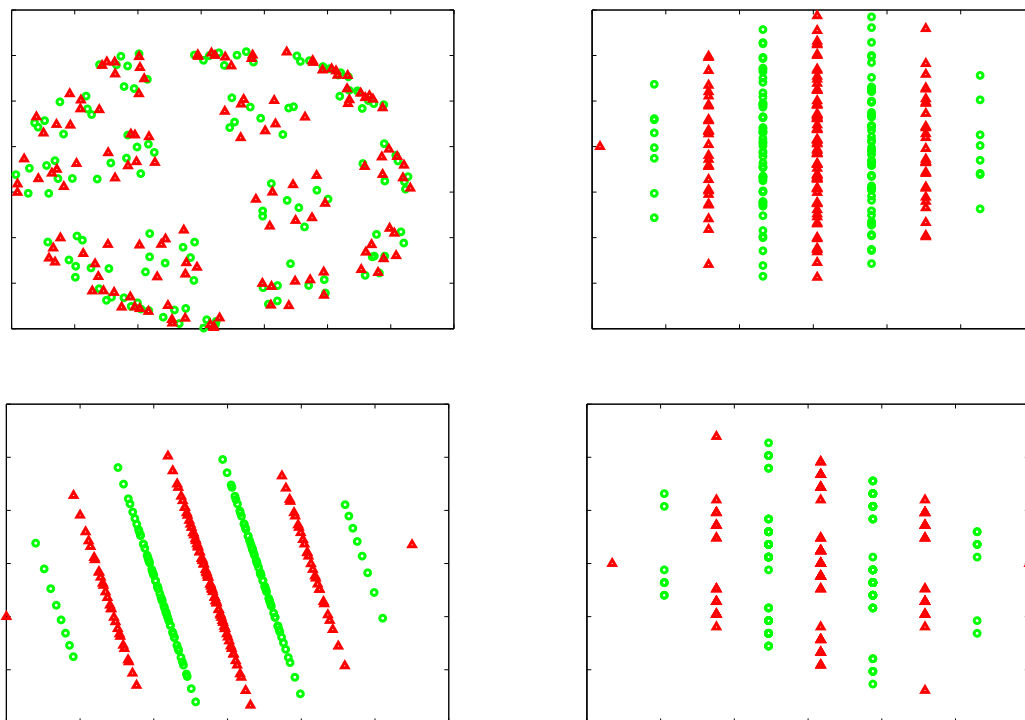
Dane *Breast-w* wyglądają podobnie (osiągają podobną separację danych) dla wszystkich porównywanych tutaj metod rzutowania (zob. rys. 4.9).

Rzutowanie za pomocą SVM pokazuje znaczącą separację dla *Leukemii* (rys. 4.10), wynikającą zapewne z maksymalizacji marginesu. Pierwszy kierunek pokazuje drobne nakładanie się danych, ale w testach przy użyciu krosvalidacji okazuje się on być znacznie lepszym niż drugi kierunek.

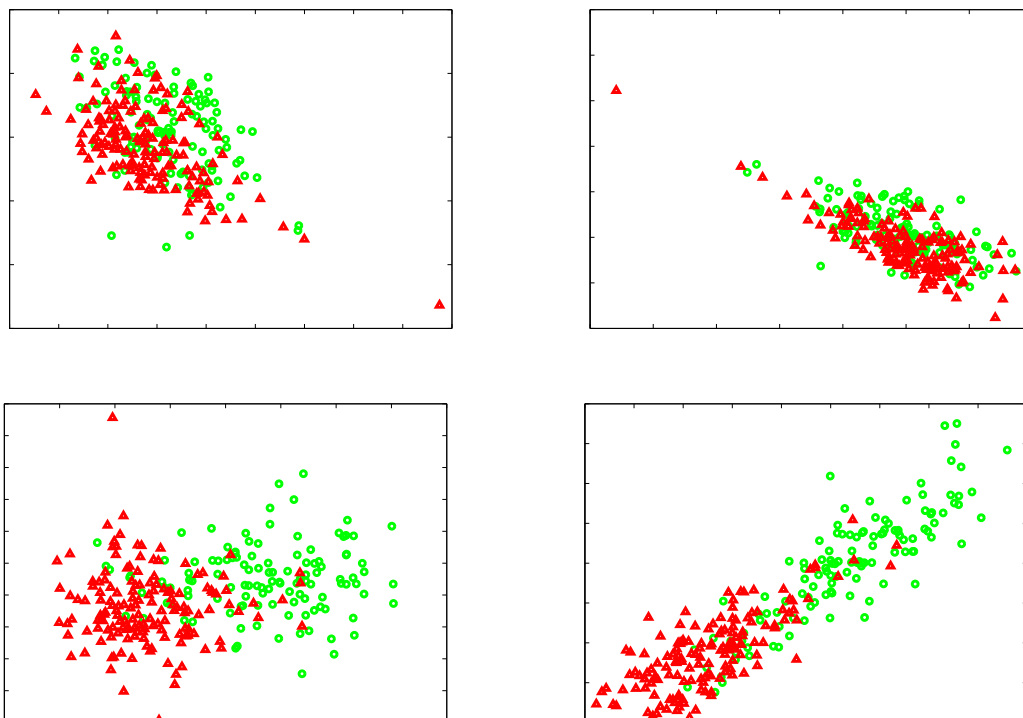
W celu porównania wpływu redukcji wymiarowości na dokładność klasyfikacji przy pomocy SVMa z liniowym kernelem, dla każdego ze zbiorów danych dokonano 10 razy



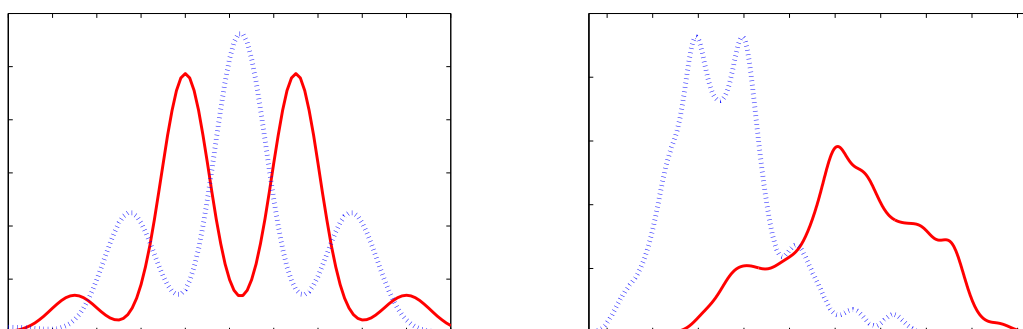
Rysunek 4.5: Porównanie trzech typów inicjalizacji MDS (zbiór *Wine*): linia niebieska ciągła - losowa, linia zielona kropkowana - PCA, linia czerwona przerywana - TriVis.



Rysunek 4.6: Zbiór *Parity8*; górny wiersz: MDS i PCA, dolny wiersz: FDA i SVM.



Rysunek 4.7: Zbiór *Heart-c*; górny wiersz: MDS i PCA, dolny wiersz: FDA i SVM.

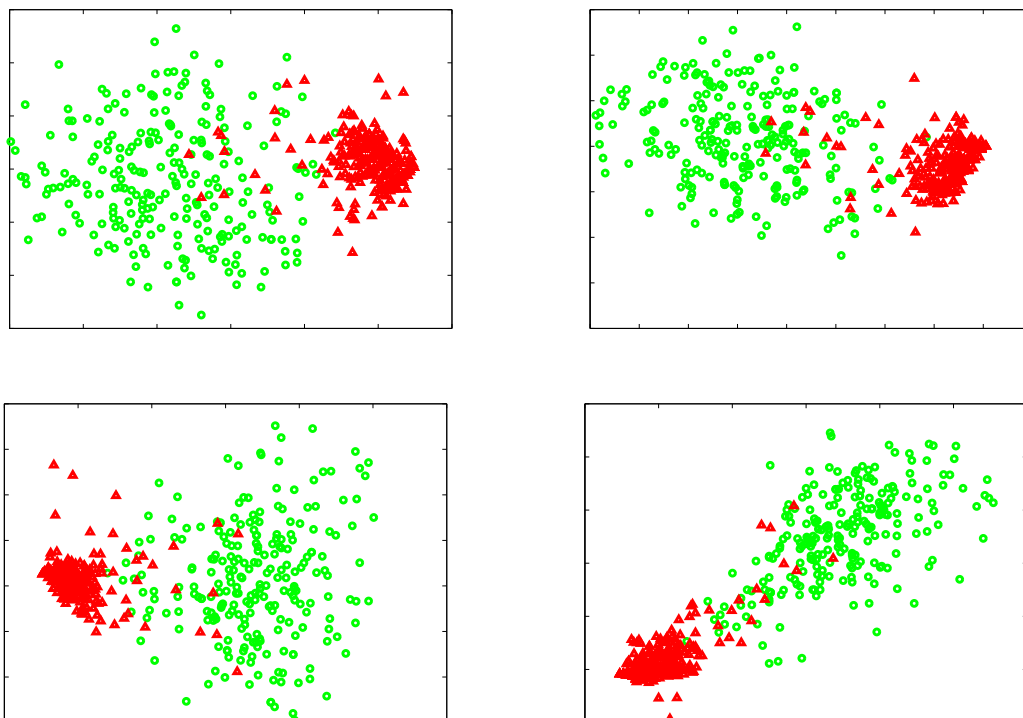


Rysunek 4.8: Estymacja prawdopodobieństwa w zależności od klasy dla zbiorów *Parity8* i *Heart-c* w pierwszym kierunku SVM.

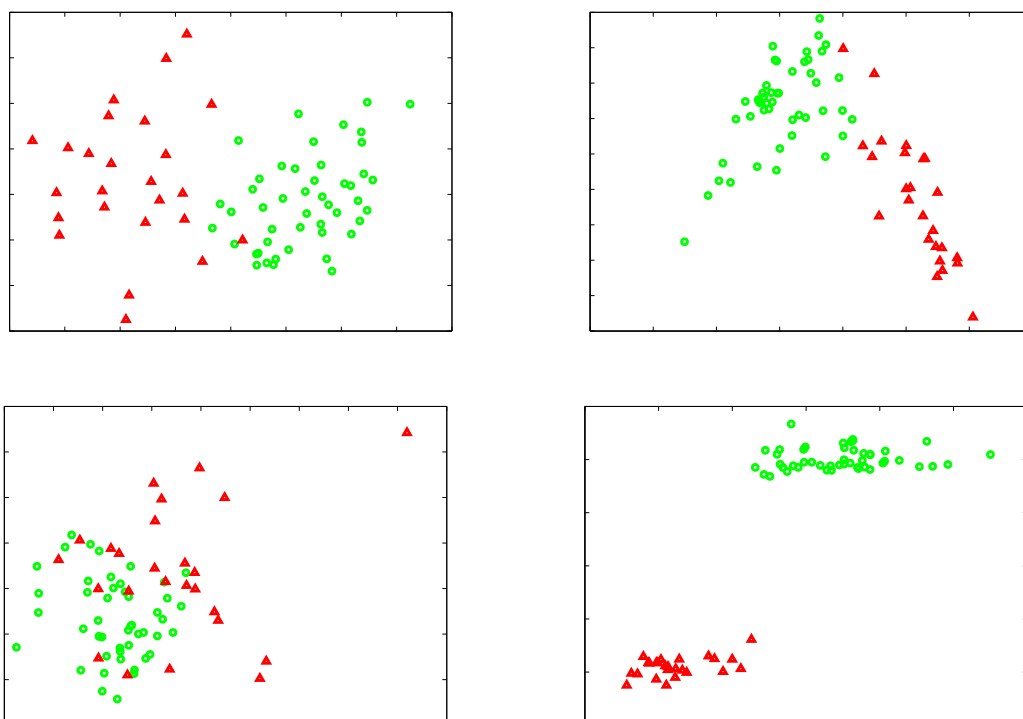
powtórzonej 10-krotnej krosvalidacji, a uśrednione wyniki zaprezentowano w tabeli 4.1 z zanotowanymi poprawnościami klasyfikacji i odchyleniami standardowymi. Testy wykonano zarówno na zbiorach oryginalnych, jak i na zmapowanych do jednego i dwóch wymiarów.

W przypadku *Parity8* widać, że poprawności klasyfikacji są podobne dla każdej z użytych metod. Wynika to z faktu, który został przytoczony nieco wcześniej - że dane nie są separowalne przy pomocy klasyfikatora liniowego. Dla zbioru *Heart-c* i *Breast-w*, rzutowania FDA i SVM dają zbliżone do siebie wyniki, istotnie lepsze niż pozostałe metody. Podobnie zachowuje się zbiór *Leukemia* - tutaj rzutowanie SVM działa najlepiej.

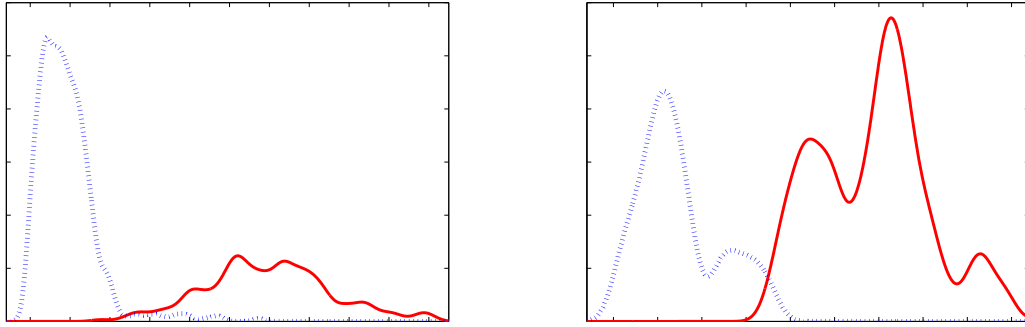
Warto zauważyć, że w większości przypadków już mapowanie do 1 wymiaru okazuje się być wystarczającym. W przypadku rzutowania SVM, dla każdego z testowanych



Rysunek 4.9: Zbiór *Breast-w*; górny wiersz: MDS i PCA, dolny wiersz: FDA i SVM.



Rysunek 4.10: Zbiór *Leukemia*; górny wiersz: MDS i PCA, dolny wiersz: FDA i SVM.



Rysunek 4.11: Estymacja prawdopodobieństwa w zależności od klasy dla zbiorów *Breast-w* i *Leukemia* w pierwszym kierunku SVM.

	Liczba cech	Parity8	Heart-c	Breast-w	Leukemia
PCA	1	41.8 ± 6.2	55.6 ± 8.3	65.0 ± 6.0	65.2 ± 15.6
PCA	2	41.7 ± 5.3	55.6 ± 8.3	65.0 ± 6.0	77.6 ± 19.1
MDS	1	39.7 ± 5.8	60.3 ± 9.3	97.0 ± 2.0	60.2 ± 18.1
MDS	2	38.2 ± 5.4	68.6 ± 9.0	96.7 ± 2.1	94.5 ± 8.4
FDA	1	40.3 ± 6.5	85.0 ± 6.6	97.2 ± 1.9	75.6 ± 15.4
FDA	2	38.7 ± 7.1	85.2 ± 6.3	97.1 ± 2.0	81.8 ± 14.1
SVM	1	41.9 ± 6.5	84.8 ± 6.5	97.3 ± 1.8	97.2 ± 5.7
SVM	2	41.8 ± 6.2	84.8 ± 6.5	97.3 ± 1.8	97.2 ± 5.7
	wszystkie	31.4 ± 4.8	83.9 ± 6.3	96.6 ± 2.0	95.4 ± 7.8

Tabela 4.1: Wyniki SVM z 10x10-krotnej krosvalidacji dla danych ze zredukowaną liczbą cech.

tutaj zbiorów, nie widać poprawy jakości klasyfikacji przy używaniu drugiego kierunku. Oczywiście nie znaczy to iż nie okaże się on (a także kolejne kierunki) pomocny w przypadku znacznie trudniejszych zbiorów danych.

Należy pamiętać tutaj o tym, że rysunki nie do końca odzwierciedlają to, co otrzymujemy z wyników krosvalidacji. Wynika to z faktu, iż dane wizualizowane są w całości, natomiast przy testach używane są podzbiory danego zbioru danych.

Rozdział 5

Techniki klasyfikacji

W niniejszej pracy pominięto opisy m.in. systemów opartych na logice rozmytej, klasyfikatorów regułowych oraz takich technik uczenia maszynowego jak klasteryzacja, selekcja prototypów. Nie przedstawiono również rozmaitych mechanizmów meta-uczenia, których najprostsze elementy to Stacking czy Boosting. Uczyniono to ze względu na dość znikomy ich wpływ na zaprezentowane w rozprawie algorytmy i nie występuje potrzeba ich opisu jako tła dla proponowanych rozwiązań. Bardzo dobrymi książkami dotyczącymi dziedziny uczenia maszynowego są [31, 91], w których m.in. bardziej szczegółowo przedstawiono opisywane w niniejszej rozprawie algorytmy. Odnosnie meta-uczenia sztandarowym źródłem wiedzy wydaje się być [49], w której opisano wiele istniejących jak i zaproponowano szereg nowych mechanizmów.

5.1 Klasyfikacja danych

Klasyfikacja danych lub rozpoznawanie wzorców (*pattern recognition*) to proces składający się z dwóch etapów - uczenia oraz testu. Proces uczenia klasyfikatora polega na adaptacji jego parametrów α w celu minimalizacji określonej funkcji błędu. Celem uczenia, oprócz dokładności dopasowania do danych treningowych, jest jakość generalizacji modelu, a więc jakość klasyfikacji na danych, które nie zostały użyte podczas uczenia modelu (czyli na danych testowych). Uczenie klasyfikatora powinno być wykonywane tylko na próbce reprezentatywnej tzn. rozkład $p(x|C)$ powinien być taki sam jak reszty zbioru danych, a każdy element próbki powinien być pobierany w sposób losowy. Do oceny jakości modelu służy zbiór testowy, a proces ten nazywany jest testowaniem klasyfikatora. W procesie tym modelowi prezentowany jest zbiór testowy, a jego zadaniem jest wyznaczenie wartości wyjściowych, czyli etykiet klas dla wszystkich elementów tego zbioru. Ocena jakości działania klasyfikatora sprowadza się zwykle do wyznaczenia współczynników określających dokładność acc (lub błąd err) klasyfikacji:

$$acc = \frac{m^{pop}}{m} \quad (5.1)$$

$$err = 1 - acc \quad (5.2)$$

gdzie:

m^{pop} – liczba poprawnie sklasyfikowanych przypadków zbioru testowego,

m – całkowita liczba przypadków zbioru testowego.

Jednakże istnieją też inne, często lepsze miary poprawności klasyfikacji. W przypadku gdy mamy do czynienia ze zbiorem niezbalansowanym, przedstawione powyżej miary zafałszowują zdolność klasyfikacji. Bardziej poprawnym okazuje się być tutaj zbalansowana dokładność klasyfikacji uwzględniająca proporcje klas:

$$bacc = \frac{1}{c} \sum_{i=1}^c \left(\frac{m_i^{pop}}{m_i} \right) \quad (5.3)$$

Współczynnik ten wierniej oddaje realną dokładność klasyfikacji, traktując wszystkie klasy równomiernie, niezależnie od wewnętrznego rozkładu liczby przypadków w poszczególnych klasach.

Istnieją także inne miary poprawności klasyfikacji, oparte o tzw. macierz konfuzji. Przykład takiej macierzy dla problemów dwuklasowych pokazano w tabeli 5.1. Dla macierzy tej pożądanym jest by elementy znajdujące się poza główną przekątną dążyły do zera.

klasa/prognoza	pozytywne	negatywne
pozytywne	prawdziwie pozytywne (PP)	fałszywie negatywne (FN)
negatywne	fałszywie pozytywne (FP)	prawdziwie negatywne (PN)

Tabela 5.1: Macierz konfuzji Q dla problemów dwuklasowych.

$$Q = \begin{bmatrix} PP & FN \\ FP & PN \end{bmatrix} \quad (5.4)$$

Wówczas dokładność klasyfikacji wyznacza się jako:

$$acc = \frac{PP + PN}{PP + FN + FP + PN} \quad (5.5)$$

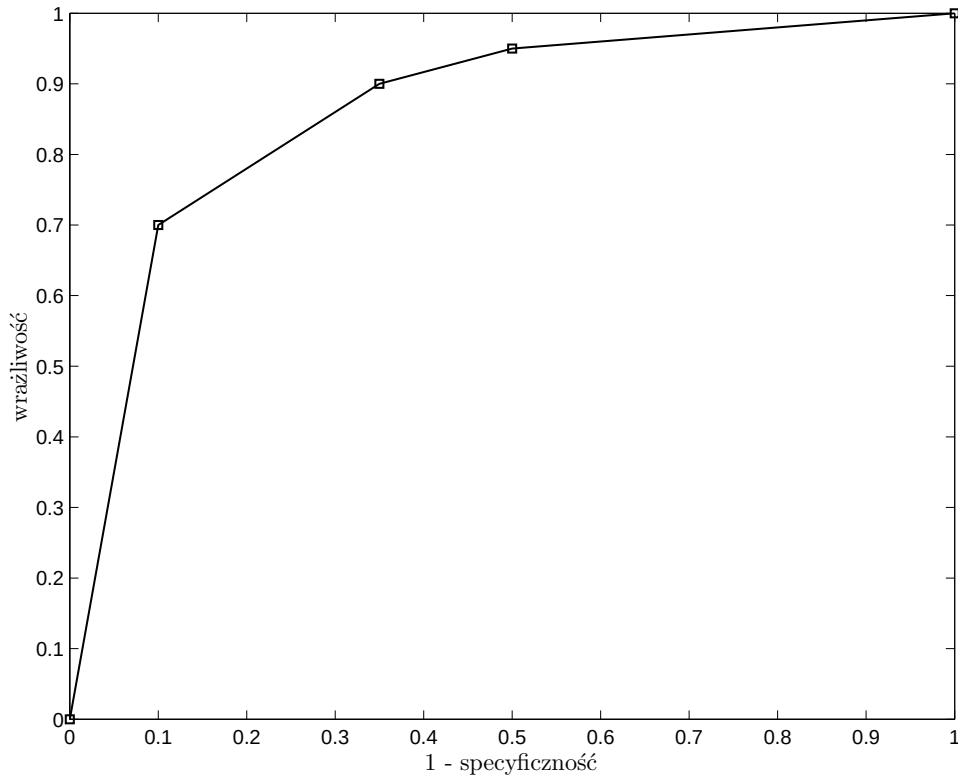
Inne, użyteczne miary to:

$$wrażliwość = \frac{PP}{PP + FN} \quad (5.6)$$

$$specyficzność = \frac{PN}{FP + PN} \quad (5.7)$$

Warto zwrócić uwagę, że nie koniecznie najlepszym dla danego problemu jest maksymalizacja poprawności klasyfikacji. Może okazać się że bardziej pożądane dla optymalizacji będzie wzięcie z różną wagą różnych elementów macierzy konfuzji. Dzięki takiemu rozwiązaniu można uniknąć błędów w wybranych obszarach macierzy konfuzji. Może być to niezwykle przydatne np. w różnych problemach medycznych, w których często dąży się do minimalizacji liczby elementów fałszywie pozytywnych.

W większości wypadków, gdy niezbędne jest określenie dopuszczalnego błędu klasyfikacji poszczególnych klas, korzysta się z charakterystyki ROC (*receiver operating characteristic*). Krzywą ROC wykreśla się jako zależność dopełnienia specyficzności w funkcji wrażliwości. Dzięki temu możliwe jest wyznaczenie optymalnego progu klasyfikacji poszczególnych klas. Zostało to przedstawione na rysunku 5.1. Na podstawie krzywej ROC można także wyznaczyć pole pod krzywą, zwane współczynnikiem AUC (*area under curve*). AUC informuje, który z klasyfikatorów jest najdokładniejszy spośród wszystkich zbadanych. Im większe pole pod krzywą, tym lepszy klasyfikator.



Rysunek 5.1: Przykład krzywej ROC.

Najczęstszym sposobem testowania poprawności maszyn uczących jest krosvalidacja (aby wynik był wiarygodny - najlepiej wielokrotnie powtórzona). Technika ta polega na k -krotnym powtórzeniu uczenia maszyny i jej testowaniu na specjalnie spreparowanych danych, na podstawie jednego zbioru danych $\mathbf{X}$. Najpierw dzieli się zbiór $\mathbf{X}$ na k możliwie równolicznych podzbiorów: $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k$, w których dodatkowo w każdym z podzbiorów $\mathbf{X}_i$ stosunki liczby wektorów z danych klas są możliwie bliskie ich estymowanym prawdopodobieństwom apriorycznym w zbiorze $\mathbf{X}$ (tzw. krosvalidacja stratyfikowana). Dodatkowo wykorzystywane będą także zbiory:

$$\mathbf{X}'_i = \mathbf{X} \setminus \mathbf{X}_i \quad (5.8)$$

a więc dopełnienia zbioru $\mathbf{X}_i$ względem $\mathbf{X}$. Części $\mathbf{X}_i$ są nazywane częściami testowymi, a części $\mathbf{X}'_i$ częściami uczącymi. Następnym krokiem krosvalidacji jest uczenie i testowanie wybranej maszyny, w celu uzyskania poprawności klasyfikacji na poszczególnych częściach testowych $\mathbf{X}_i$. Można zatem wyznaczyć średnią dokładność klasyfikacji:

$$acc = \frac{1}{k} \sum_{i=1}^k acc_i \quad (5.9)$$

oraz odchylenie standardowe:

$$std = \frac{1}{k-1} \sum_{i=1}^k (acc_i - \overline{acc})^2 \quad (5.10)$$

5.1.1 Eliminatory

Działanie algorytmów klasyfikacji może nie być wiarygodne z wielu powodów: zaszumionych danych, niewystarczającej ilości informacji, itd. Jednakże w pewnych sytuacjach mogą się one okazać użyteczne, jeżeli użyjemy ich (w odróżnieniu od klasyfikacji) do eliminacji tych klas, które dla danego przypadku są niewłaściwe. Tak więc alternatywą do klasyfikacji jest eliminacja niewłaściwych klas, przewidując że nieznany przypadek należy do podgrupy k z ogólnego zbioru K możliwych klas. Liczba k nie powinna być stała, jako że w pewnych fragmentach przestrzeni danych wejściowych różne klasy mogą się nakładać na siebie w różnym stopniu i w inny sposób. Takie rozwiązanie zostało nazwane *eliminatorami* [18], ponieważ jego podstawowym zadaniem jest eliminacja z wysokim poziomem zaufania tych klas, które są niewłaściwe. Dowolny model M , który estymuje prawdopodobieństwa klasyfikacji $p(C_i|\mathbf{X}; M)$ może zostać użyty do utworzenia nowych klas lub do eliminacji pewnych klas, przewidując że $\mathbf{X}$ należy do dwóch lub więcej klas.

5.2 Naiwny klasyfikator Bayesa

Podstawą działania tego algorytmu jest ogólnie znane twierdzenie Bayesa:

$$P(c|\mathbf{X}) = \frac{P(c)P(\mathbf{X}|c)}{P(\mathbf{X})} \quad (5.11)$$

Określa ono prawdopodobieństwo wystąpienia klasy c (spośród zbioru wszystkich klas $\mathbf{C}$), pod warunkiem zaobserwowania danych $\mathbf{X}$. $P(c)$ opisuje prawdopodobieństwo *a priori* wystąpienia klasy c , $P(\mathbf{X})$ to prawdopodobieństwo zaobserwowania danych $\mathbf{X}$, natomiast $P(\mathbf{X}|c)$ to prawdopodobieństwo wystąpienia klasy c pod warunkiem wystąpienia danych $\mathbf{X}$.

Naiwny klasyfikator Bayesa zakłada, że zmienne odpowiadające poszczególnym wymiarom przestrzeni cech są niezależne, a klasyfikacja przypadków opiera się na prawdopodobieństwie ich przynależności do poszczególnych klas. Wyznaczenie klasy, dla której prawdopodobieństwo jest największe uzyskuje się w następujący sposób:

$$NBC(x') = \arg \max_{c \in \mathbf{C}} P(c) \prod_{i=1}^n P(x'_i|c) \quad (5.12)$$

Stosuje się go najczęściej dla danych dyskretnych (jeśli pewne cechy są ciągłe, to poddaje się je dyskretyzacji). Jednakże zakładając normalność rozkładu każdej z cech dla każdej klasy, można w prosty sposób oszacować prawdopodobieństwo cech z wartościami ciągłymi:

$$P(x_i|c) = G(x, \mu_i^c, \sigma_i^c) \quad (5.13)$$

gdzie:

G – gęstość rozkładu normalnego,

μ_i^c – wartość średnia i -tej cechy dla klasy c ,

σ_i^c – odchylenie standardowe i -tej cechy dla klasy c .

Postawione założenia co do niezależności poszczególnych cech oraz ich rozkładu normalnego w praktyce mogą okazać się błędne, wówczas klasyfikator staje się daleki od optymalnego.

5.3 Metoda k -najbliższych sąsiadów

W metodzie k -najbliższych sąsiadów na początku szukane jest k najbliższych sąsiadów danego przypadku, a następnie przypadek ten jest przypisywany do tej klasy, do której należy większość z ustalonych jego k najbliższych sąsiadów. Zatem metoda ta w głównej mierze opiera się na obliczaniu odległości pomiędzy poszczególnymi przypadkami w zbiorze danych. Przy ich obliczaniu możemy się posłużyć różnymi miarami, z czego najczęściej wykorzystywana to miara euklidesowa. Poniżej przedstawiono kilka najbardziej popularnych miar odległości wykorzystywanych w metodzie kNN:

1. miara Euklidesa

$$D(x, y) = \sqrt{\sum_{i=1}^n f(x_i, y_i)} \quad (5.14)$$

2. miara Canberra

$$D(x, y) = \sum_{i=1, x_i+y_i \neq 0}^n \frac{|x_i - y_i|}{|x_i + y_i|} \quad (5.15)$$

3. miara Manhattan

$$D(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (5.16)$$

4. miara Chebyshev

$$D(x, y) = \max |x_i - y_i| \quad (5.17)$$

Ogromne znaczenie w przypadku tej metody ma wybór odpowiedniego k , gdyż jest to podstawowy parametr decydujący o jakości predykcji. Powinien on być na tyle mały aby k najbliższych sąsiadów było wystarczająco bliskimi sąsiadami nowego punktu, natomiast na tyle duży aby zminimalizować prawdopodobieństwo błędnych klasyfikacji. Najprościej dobór odpowiedniego k zrealizować stosując krosvalidację.

Jest to tylko najprostsza wersja algorytmu kNN, możliwe są naturalnie różne rozwinięcia jak ważenie poszczególnych wymiarów itp. [77]

5.4 Drzewa decyzyjne

Algorytmy drzew decyzyjnych [71] są wciąż podstawą większości dużych systemów uczenia maszynowego, ponieważ oferują łatwy i obliczeniowo efektywny sposób wydobywania reguł decyzyjnych z danych [21]. Drzewa oparte są o rekurencyjny podział danych i w przeciwieństwie do większości algorytmów uczenia, używają innego zestawu cech w różnych częściach przestrzeni cech, automatycznie dokonując lokalnej selekcji.

Algorytm C4.5 generowania drzew decyzyjnych [70] to powstała na bazie metody ID3 najbardziej popularna wersja, cechująca się wieloma użytecznymi własnościami. Zasada działania tej metody opiera się na podziale danego węzła na tyle podwęzłów ile wartości ma najbardziej informatywna cecha (zapewniająca maksymalną redukcję entropii). Wiąże się to niestety z tym, że algorytm ten ma tendencję do częstszego używania cech, które mają dużą (w porównaniu do innych) liczbę możliwych wartości. Aby zniwelować część słabych stron opisywanego algorytmu, wprowadzono w nim kilka zmian, takich jak modyfikacja miary nieczystości węzłów, możliwość wykorzystania cech z wartościami

ciągłymi, zmiana sposobu przycinania czy możliwość użycia danych z brakującymi wartościami dla atrybutów. W celu wyeliminowania niepożądanego efektu preferowania cech o dużej liczbie wartości, algorytm ten do wyznaczania poziomu niejednorodności węzłów używa, w odróżnieniu od zysku informacyjnego (*information gain*), miary zwanej zyskiem względnym (*gain ratio*).

W przypadku cech ciągłych rozpatrywane zostają wszystkie możliwe podziały na dwa podzbiory, zdeterminowane punktem podziału. Algorytm ten ocenia jakość dla każdego z możliwych podziałów uzyskując wartość względnego zysku informacyjnego, a następnie wybiera ten, który taki zysk maksymalizuje.

Przycinanie drzewa oparte jest na ocenie statystycznej istotności różnicy błędu klasyfikacji dla danego węzła i jego podwęzłów. Oszacowane zostaje prawdopodobieństwo zwiększenia błędu, a następnie obcięte zostają te elementy, dla których to prawdopodobieństwo nie przekracza zadanego progu.

5.4.1 Porównanie użycia różnych entropii w drzewach

Entropia jest miarą stopnia nieuporządkowania układu. Claude Shannon zdefiniował miarę entropii w następujący sposób [79]:

$$S = - \sum_{i=1}^n p_i \log_2 p_i \quad (5.18)$$

gdzie p_i to prawdopodobieństwo wystąpienia zdarzenia (wartości cechy) x_i będącego elementem zdarzenia (cechy) $\mathbf{X}$, które może przybierać różne wartości $\{x_1 \dots x_n\}$.

Entropia Shannona jest malejącą funkcją rozproszenia zmiennej losowej, która maksimum osiąga wówczas, gdy wszystkie wyniki są jednakowo prawdopodobne. Może ona zostać użyta globalnie - dla całości danych, albo lokalnie w celu wyznaczenia entropii rozkładu gęstości prawdopodobieństwa w niektórych punktach. Entropia może zostać uogólniona w celu dostarczenia dodatkowej informacji o istotności pewnych zdarzeń.

Constantino Tsallis [87] i Alfred Renyi [72] zaproponowali uogólnione wersje entropii, które dla $\alpha = 1$ redukują się do entropii Shannona.

Entropia Renyiego jest zdefiniowana jako:

$$I_\alpha = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^n p_i^\alpha \right) \quad (5.19)$$

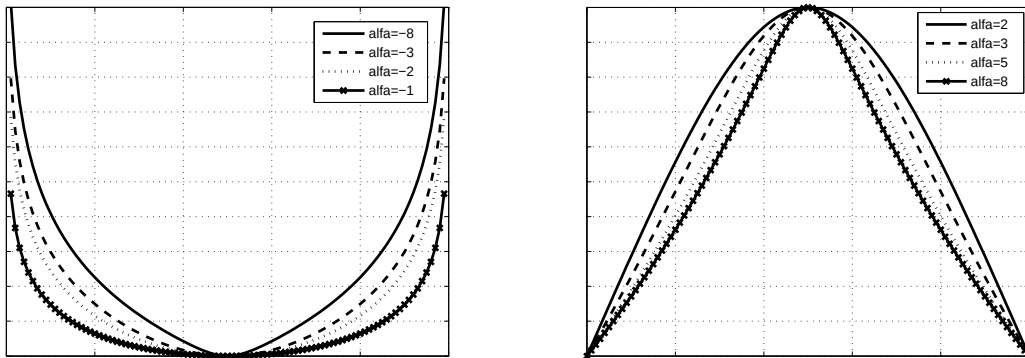
Podobnie jak entropia Shannona osiąga ona maksimum równe $\ln(n)$ dla $p_i = 1/n$, jednakże zawiera dodatkowy parametr α , który może zostać użyty w celu zwiększenia lub zmniejszenia wrażliwości na kształt rozkładów prawdopodobieństw.

Tsallis zdefiniował swoją entropię jako:

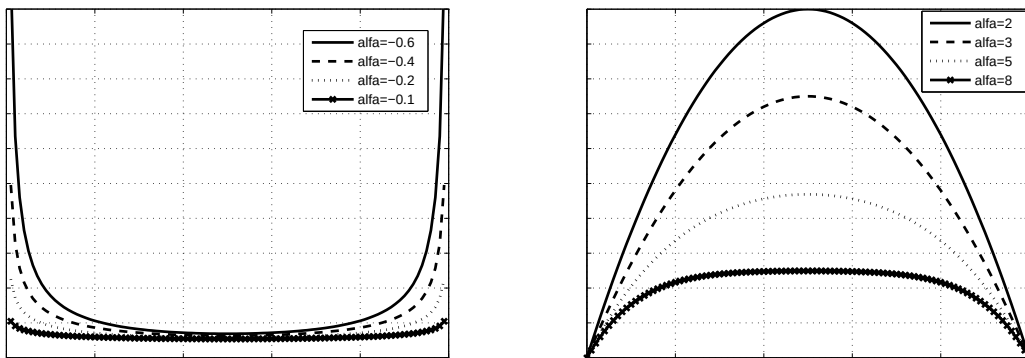
$$S_\alpha = \frac{1}{\alpha-1} \left(1 - \sum_{i=1}^n p_i^\alpha \right) \quad (5.20)$$

Rysunki 5.2 i 5.3 ilustrują porównanie entropii Renyiego, Tsallisa i Shannona dla dwóch prawdopodobieństw p_1 i p_2 gdzie $p_1 = 1 - p_2$.

Modyfikacja standardowego algorytmu C4.5 polegała na zamianie miary Shannona na jedną z dwóch opisywanych wyżej entropii. Celem tego było sprawdzenie ich wpływu



Rysunek 5.2: Wykresy entropii Renyiego dla kilku ujemnych i dodatnich wartości parametru α .



Rysunek 5.3: Wykresy entropii Tsallisa dla kilku ujemnych i dodatnich wartości parametru α .

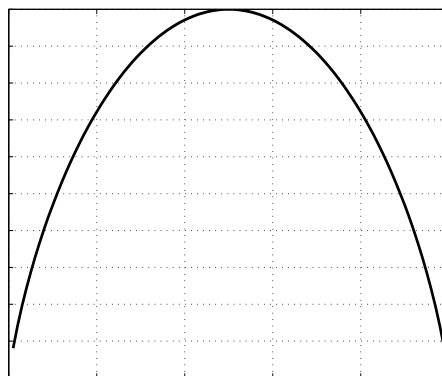
na własności drzew decyzyj. Ostateczne kryterium podziału oparte było na mierze zysku względnego, tak więc test atrybutu A , który dzieli dane $\mathbf{D}$ na dwie części $\mathbf{D}_t$ i $\mathbf{D}_f$ (przy zbiorze klas ω) daje zysk:

$$G(\omega, A|\mathbf{D}) = H(\omega|\mathbf{D}) - \frac{|\mathbf{D}_t|}{|\mathbf{D}|}H(\omega|\mathbf{D}_t) - \frac{|\mathbf{D}_f|}{|\mathbf{D}|}H(\omega|\mathbf{D}_f) \quad (5.21)$$

gdzie $|\mathbf{D}|$ to liczba elementów w zbiorze $\mathbf{D}$, a $H(\omega|S)$ to jedna z trzech entropii: Shannona, Renyiego albo Tsallisa.

Poniżej przedstawiono wyniki pierwszych eksperymentów wykonanych przy użyciu drzewa C4.5 ze zmodyfikowanym kryterium podziału. Wyniki przedstawione w tabelach 5.2-5.7 wyraźnie pokazują, że dodatkowy parametr α ułatwia klasyfikację danych z jedną lub kilkoma mało licznymi klasami. Wartość tego współczynnika może być łatwo wyznaczona poprzez wewnętrzną krosvalidację.

Dla każdego zbioru danych przeprowadzono 10 razy powtórzoną 10-krotną krosvalidację, testując zarówno standardowe drzewo C4.5, jak i jego zmodyfikowane wersje z entropiami Tsallisa i Renyiego. Testy wykonano dla kilku wartości parametru α , a wyniki (średnie dokładności klasyfikacji i odchylenia standardowe) zanotowano w tabelach 5.2-5.7.



Rysunek 5.4: Wykres entropii Shannona.

Entropia	Alfa							
	-1.5	-0.9	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	64.6±0.2	64.6±0.2	64.6±0.2	77.3±4.1	75.4±2.1	77.7±3.3	77.8±4.7	79.1±2.6
Tsallis	64.6±0.2	64.6±0.2	64.6±0.2	64.6±0.2	77.3±3.7	75.4±4.0	74.4±4.3	71.3±5.4
Entropia	Alfa							
	0.9	1.1	1.3	1.5	2.0	3.0	4.0	5.0
Renyi	78.8±4.4	82.1±4.2	82.8±4.0	82.9±2.5	84.0±3.9	79.4±3.0	80.8±3.1	78.9±2.2
Tsallis	73.0±3.4	74.9±1.8	73.4±2.4	71.1±4.0	70.2±3.9	73.9±4.4	72.8±3.6	71.1±4.4
Shannon	81.2±3.7							

Tabela 5.2: Wyniki poprawności klasyfikacji dla zbioru *Colon-cancer*.

Entropia	Klasa	Alfa						
		-1.5	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	1	0.0±0.0	0.0±0.0	59.7±4.7	58.7±6.8	60.8±6.4	63.2±7.3	66.0±6.5
	2	100±0.0	100±0.0	87.2±4.1	84.7±2.4	87.2±2.8	85.8±4.0	86.2±3.9
Tsallis	1	0.0±0.0	0.0±0.0	0.0±0.0	58.2±5.1	59.8±9.3	59.8±5.2	50.7±10.2
	2	100±0.0	100±0.0	100±0.0	87.6±4.8	83.9±4.3	82.8±4.4	82.6±3.9
Entropia	Klasa	Alfa						
		0.9	1.1	1.5	2.0	3.0	4.0	5.0
Renyi	1	65.8±6.5	70.0±7.2	67.3±5.4	69.2±6.6	58.5±2.9	61.0±3.8	58.7±3.5
	2	85.8±4.6	88.8±4.5	91.5±2.1	92.1±2.9	90.7±4.3	91.6±3.7	90.1±3.6
Tsallis	1	55.7±7.7	58.5±4.9	58.3±8.2	53.2±7.6	67.2±9.2	60.5±7.4	60.0±10.4
	2	82.8±3.4	84.2±2.2	78.9±4.6	80.0±3.7	77.7±6.0	79.7±5.3	77.3±3.4
Shannon	1	69.5±4.2						
	2	87.7±4.8						

Tabela 5.3: Wyniki poprawności klasyfikacji każdej z klas dla zbioru *Colon-cancer*.

Jak łatwo zauważyć, najlepsza wartość parametru α jest różna dla różnych zbiorów, ale może być ona łatwo ustalona przy pomocy krosvalidacji. Warto także zauważyć, że nawet jeśli zmiana wyżej wymienionego parametru nie spowoduje wzrostu poprawności klasyfikacji, to może się okazać, że przyniesie wzrost wrażliwości lub specyficzności (co jest często pożądane np. w przypadku problemów medycznych).

Po przeprowadzonych eksperymentach wyraźnie widać, że wyniki ściśle zależą od wartości parametru α . Jeżeli $\alpha = 1$, wówczas drzewo ze zmodyfikowaną entropią osiąga rezultaty tożsame z drzewem używającym entropii Shannona.

Entropia	Alfa							
	-1.5	-0.9	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	46.0±4.2	46.0±4.2	46.0±4.2	69.9±5.2	71.8±5.4	70.7±5.4	70.5±5.0	73.0±4.9
Tsallis	52.4±6.8	52.4±6.8	52.4±6.8	52.4±6.8	71.1±5.6	69.8±5.2	72.4±6.0	79.9±5.0
Entropia	Alfa							
	0.9	1.1	1.3	1.5	2.0	3.0	4.0	5.0
Renyi	76.5±6.7	81.0±6.2	81.0±4.8	80.5±5.0	79.3±5.1	79.5±5.6	75.9±7.2	69.7±6.3
Tsallis	81.3±4.7	82.0±4.3	81.8±5.2	80.8±6.5	81.5±5.7	78.8±6.9	81.8±4.1	80.5±4.0
Shannon	78.5±4.8							

Tabela 5.4: Wyniki poprawności klasyfikacji dla zbioru *DLBCL*.

Entropia	Klasa	Alfa						
		-1.5	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	1	90.0±10.5	90.0±10.5	72.3±8.4	75.5±10.6	74.3±11.2	76.3±8.7	79.7±7.1
	2	10.0±10.5	10.0±10.5	65.5±11.8	66.7±10.5	65.7±10.9	62.5±8.5	65.3±5.9
Tsallis	1	64.8±10.6	64.8±10.6	64.8±10.6	74.5±12.1	73.3±10.7	80.2±8.5	85.8±7.2
	2	41.8±11.0	41.8±11.0	41.8±11.0	65.7±10.6	65.2±12.0	64.8±9.1	74.7±9.8
Entropia	Klasa	Alfa						
		0.9	1.1	1.3	1.5	2.0	3.0	5.0
Renyi	1	82.7±8.4	85.5±8.1	86.5±5.8	85.2±5.8	84.8±6.4	84.2±5.3	68.0±12.0
	2	70.2±11.1	77.3±9.0	77.0±7.4	77.3±7.6	74.7±8.1	75.3±9.0	69.3±4.7
Tsallis	1	88.2±6.3	88.2±5.7	86.2±5.3	85.2±6.4	84.7±5.7	83.2±8.8	87.3±5.2
	2	76.0±7.5	77.3±5.3	78.7±6.9	77.8±9.3	80.0±6.9	76.3±6.4	75.3±4.9
Shannon	1	84.8±7.0						
	2	72.7±8.7						

Tabela 5.5: Wyniki poprawności klasyfikacji każdej z klas dla zbioru *DLBCL*.

Entropia	Alfa						
	-1.5	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	65.4 ± 0.4	65.4 ± 0.4	88.5 ± 2.4	85.6 ± 3.9	84.6 ± 3.8	82.4 ± 4.6	82.0 ± 4.6
Tsallis	65.4 ± 0.4	65.4 ± 0.4	65.4 ± 0.4	83.5 ± 4.4	84.8 ± 4.2	84.3 ± 3.5	82.3 ± 3.9
Entropia	Alfa						
	0.9	1.1	1.3	1.5	2.0	3.0	5.0
Renyi	80.5±3.8	81.5±3.5	82.2±3.5	82.4±2.6	85.3±2.8	86.1±2.8	83.8±2.0
Tsallis	82.5±4.4	81.5±2.9	82.3±1.1	83.3±1.4	82.2±2.5	86.5±2.7	87.5±3.6
Shannon	81.4±4.1						

Tabela 5.6: Wyniki poprawności klasyfikacji dla zbioru *Leukemia*.

Dla zbioru *Colon-cancer* (opis zbioru w dodatku A) (tabele 5.2 i 5.3) największa dokładność uzyskiwana jest dla entropii Renyiego przy $\alpha = 2$, z mniejszą wariancją oraz znacznie wyższą specyficznością (dokładność klasyfikacji drugiej klasy) od tych uzyskanych przy pomocy entropii Shannona. Entropia Tsallisa w przypadku tego zbioru najlepiej sprawuje się dla $\alpha = 1$, kiedy zrównuje się z wynikami standardowego algorytmu.

W przypadku *DLBCL* (opis zbioru w dodatku A), entropia Renyiego jak i Tsallisa uzyskuje najlepsze rezultaty dla α z przedziału 1.1 – 1.3, poprawiając zarówno czułość jak i wrażliwość (w porównaniu z entropią Shannona) (tabele 5.4 i 5.5).

Dla *Leukemii* (opis zbioru w dodatku A) najlepszy rezultat daje użycie entropii Renyiego przy $\alpha = -0.1$, oferując poprawność klasyfikacji na poziomie $88.5 \pm 2.4\%$, co jest znacząco lepszym wynikiem od uzyskanego przez entropię Shannona wynoszącym $81.4 \pm 4.1\%$.

Entropia	Klasa	Alfa						
		-1.5	-0.5	-0.1	0.1	0.3	0.5	0.7
Renyi	1	100±0.0	100±0.0	89.2±1.6	89.4±3.2	88.7±2.5	86.2±3.2	84.8±4.0
	2	0.0±0.0	0.0±0.0	86.6±5.4	77.7±9.9	75.8±10.5	74.3±10.5	76.2±10.6
Tsallis	1	100±0.0	100±0.0	100±0.0	88.3±3.6	88.7±2.9	89.0±3.3	85.4±3.2
	2	0.0±0.0	0.0±0.0	0.0±0.0	73.5±10.1	76.8±10.3	74.8±9.3	76.6±10.3
Entropia	Klasa	Alfa						
		0.9	1.3	1.5	2.0	3.0	4.0	5.0
Renyi	1	85.0±3.9	84.7±4.6	85.2±3.7	88.6±4.5	90.2±3.4	83.9±3.9	86.8±2.0
	2	71.3±11.9	77.4±5.0	76.7±5.7	79.1±4.6	78.3±3.7	80.3±7.1	78.2±5.9
Tsallis	1	84.5±3.9	86.1±3.4	87.4±4.0	85.0±3.4	90.0±3.8	89.1±3.0	91.3±3.5
	2	78.1±11.8	74.4±7.5	75.2±7.8	77.9±7.0	80.2±6.7	84.3±6.2	80.3±6.5
Shannon	1	83.8±5.3						
	2	76.6±5.7						

Tabela 5.7: Wyniki poprawności klasyfikacji każdej z klas dla zbioru *Leukemia*.

Teoretyczne rozważania i testy na sztucznych danych pokazują, że drzewo decyzyjne ze zmodyfikowanym kryterium podziału może łatwo poradzić sobie z poprawną klasyfikacją jednej lub wielu mało licznych klas. Parametr α zapewnia możliwość zwiększenia poprawności klasyfikacji klas o różnej liczebności wektorów. Sprawia to, że zmodyfikowany algorytm drzew decyzyjnych staje się bardziej atrakcyjnym w porównaniu do oryginalnej wersji opartej o entropię Shannona.

5.5 Liniowa dyskryminacja

Prawie 80 lat temu Fisher zaproponował rozwiązanie problemu klasyfikacji przy pomocy algorytmu dyskryminacji liniowej (*Linear discrimination algorithm, LDA*) [54]. W pierwszej wersji metoda ta została zaproponowana dla dwóch klas i bazowała na mechanizmie rzutowania danych na prostą. Rezultatem działania LDA jest funkcja liniowa, która w możliwie najlepszy sposób odseparowuje przykłady należące do dwóch przeciwnych klas. Funkcja ta znajdowana jest przy pomocy rzutowania danych na prostą, na której minimalizowany jest rozrzut każdej z klas, przy jednoczesnej maksymalizacji separacji pomiędzy tymi klasami. Założenia te można spełnić dzięki minimalizacji następującej funkcji:

$$J(w) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad (5.22)$$

gdzie $\mathbf{S}_W$ jest macierzą rozrzutu dla wektorów z tej samej klasy, a $\mathbf{S}_B$ macierzą rozrzutu dla wektorów różnych klas. Warto zauważyć, że macierze rozrzutu są proporcjonalne do macierzy kowariancji, dlatego możliwe jest zdefiniowanie J poprzez macierze kowariancji.

$$\mathbf{S}_B = \sum_c N_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \quad (5.23)$$

$$\mathbf{S}_W = \sum_c N_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \quad (5.24)$$

W podanych wzorach μ_c i $\bar{x}$ oznaczają odpowiednio:

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i \quad (5.25)$$

$$\bar{x} = \frac{1}{N} \sum_i x_i = \frac{1}{N} \sum_c N_c \mu_c \quad (5.26)$$

gdzie N_c oznacza liczbę przykładów w klasie c . Dla problemów dwuklasowych komponent $\mathbf{S}_B$ można zdefiniować jako:

$$\mathbf{S}'_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \quad (5.27)$$

natomiast główne rozwiązanie przyjmuje postać:

$$w = \mathbf{S}_W^{-1}(\mu_1 - \mu_2) \quad (5.28)$$

Najbardziej zaawansowanym rozwinięciem idei LDA są maszyny wektorów wspierających opisane w rozdziale 5.7.

5.6 Sztuczne sieci neuronowe

Każda sieć neuronowa zbudowana jest z elementów zwanych umownie neuronami, które przetwarzają sygnały wejściowe w sygnał wyjściowy. Przykładowy model neuronu pokazano na rysunku 5.5. Każdy neuron charakteryzuje się tym, że ma kilka wejść i jedno wyjście. Sygnał wyjściowy Y jest funkcją sygnałów wejściowych $\mathbf{X}$ wpływających do węzła (neuronu) oraz wag, które są z nim związane:

$$Y = f(\mathbf{X}, \mathbf{W}) \quad (5.29)$$

Funkcja ta nazywana jest funkcją transferu i ma najczęściej charakter nieliniowy (w zależności od konkretnego celu, jakiemu służy dany neuron, może ona przybierać różną postać). Najprostszym przykładem funkcji transferu może być funkcja liniowa:

$$Y = \sum_i \mathbf{W}_i \mathbf{X}_i \quad (5.30)$$

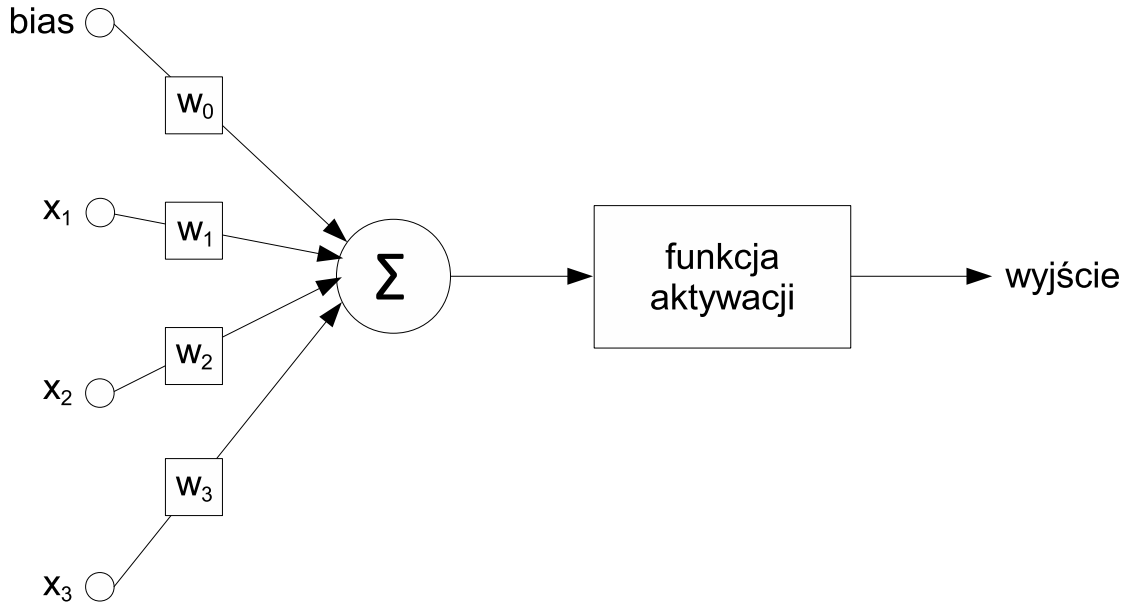
Jednakże najczęściej stosowaną w sieciach propagacji wstecznej jest nieliniowa funkcja sigmoidalna:

$$Y = \sigma(\mathbf{W}\mathbf{X} + \theta) = \frac{1}{1 + e^{-s(\sum_i \mathbf{W}_i \mathbf{X}_i + \theta)}} \quad (5.31)$$

Neurony połączone ze sobą tworzą sieć neuronową, zaś liczba i sposób połączenia neuronów określa architekturę sieci neuronowej. Neurony występujące w sieciach MLP (*Multi Layer Perceptron*) tworzą warstwy: wejściową, ukrytą i wyjściową, przy czym warstw ukrytych może być kilka. W warstwie wejściowej neurony nie wykonują żadnych obliczeń, a jedynie “pobierają” dane uczące lub testowe. Warstwa ukryta zawiera neurony, których wejścia nigdy nie łączą się bezpośrednio z sygnałem wejściowym, ani wyjściem sieci. W warstwie wyjściowej, jak nie trudno się domyśleć, występują neurony bezpośrednio połączone z wyjściem sieci. To w jaki sposób neurony zostaną połączone, a przede wszystkim ile ich będzie, determinuje kształt powstałych obszarów decyzyjnych.

Najczęściej stosowanym algorytmem uczenia sieci MLP jest jeden z wariantów algorytmu wstecznej propagacji błędów (*backpropagation*) [85, 53]. W metodzie tej uczenie polega na minimalizacji odpowiedniej funkcji kosztu, której najprostsza definicja ma postać błędu średniokwadratowego, która przy założeniu tylko jednego wyjścia sieci wygląda następująco:

$$E = \frac{1}{2} \sum_i (Y(\mathbf{X}_i, \mathbf{W}) - d_i)^2 \quad (5.32)$$



Rysunek 5.5: Model sztucznego neuronu.

Dokładny opis algorytmu wstecznej propagacji błędów zarówno w jego podstawowej postaci, jak i różnych odmianach (wsteczna propagacja z momentem, QuickProp, Rprop) można znaleźć w pracach [91, 6, 75].

W rozdziale tym opisane zostały jedynie sieci MLP, opis innych klas sieci jak sieci RBF, RAN i innych [6] został pominięty z racji ich niewielkiego związku z niniejszą rozprawą doktorską.

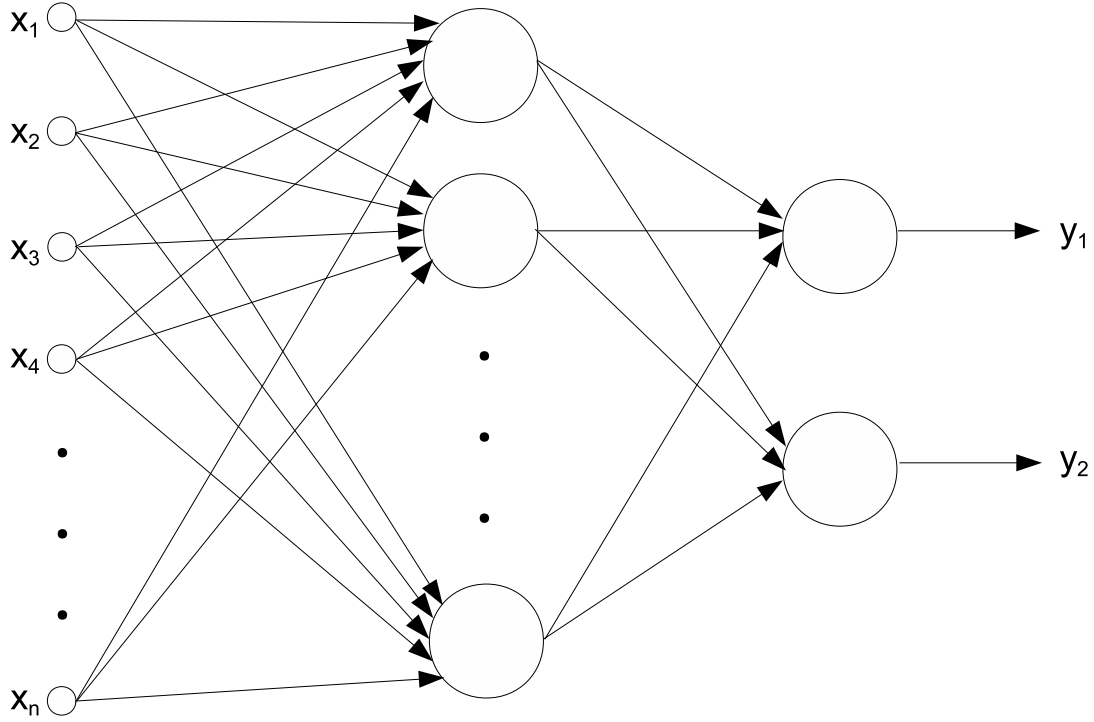
5.7 Maszyny wektorów wspierających

5.7.1 Standardowe sformułowanie SVM

Począwszy od 1992 roku, kiedy to ukazał się artykuł Bosera, Guyon i Vapnika [8] maszyny wektorów wspierających (*Support Vector Machines, SVM*) bardzo szybko stały się najbardziej popularną metodą klasyfikacji i aproksymacji. W przypadku problemów dwuklasowych SVM minimalizuje średni błąd dla wszystkich par $\langle X_i, Y_i \rangle$. W zależności od wyboru kerneli i optymalizacji ich parametrów, SVM może utworzyć odpowiednie, nieliniowe modele danych, które dzięki optymalizacji marginesu zapewniają dobrą generalizację. Oznacza to, że minimalna odległość pomiędzy wektorami treningowymi $\mathbf{X}_i$ a hiperpłaszczyzną $\mathbf{W}$ powinna zostać zmaksymalizowana:

$$\max_{\mathbf{W}, b} \min \|\mathbf{X} - \mathbf{X}_i\| : \mathbf{W} \cdot \mathbf{X} + b = 0, \quad i = 1, \dots, m \quad (5.33)$$

$\mathbf{W}$ i b mogą zostać przeskalowane w taki sposób, że punkt który znajduje się najbliżej hiperpłaszczyzny $\mathbf{W} \cdot \mathbf{X} + b = 0$ leży na jednej z równoległych hiperpłaszczyzn, definiując



Rysunek 5.6: Przykład sztucznej sieci neuronowej.

margines $\mathbf{W} \cdot \mathbf{X} + b = \pm 1$. Prowadzi to do warunku:

$$\forall_{\mathbf{X}_i} Y_i[\mathbf{W} \cdot \mathbf{X}_i + b] \geq 1 \quad (5.34)$$

Szerokość marginesu wynosi $2/\|\mathbf{W}\|$, tak więc maksymalizacja marginesu jest równoważna minimalizacji:

$$\min_{\mathbf{W}, b} \tau(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\|^2 \quad (5.35)$$

z ograniczeniami, które gwarantują poprawną klasyfikację:

$$Y_i[\mathbf{W} \cdot \mathbf{X}_i + b] \geq 1 \quad i = 1, \dots, m \quad (5.36)$$

Optymalizacja problemów z ograniczeniami jest rozwiązywana metodą Lagrange'a:

$$L(\mathbf{W}, b, \alpha) = \frac{1}{2} \|\mathbf{W}\|^2 - \sum_{i=1}^m \alpha_i (Y_i[\mathbf{X}_i \cdot \mathbf{W} + b] - 1) \quad (5.37)$$

gdzie $\alpha_i > 0$ to mnożniki Lagrange'a. Jego minimalizacja przez b i $\mathbf{W}$ prowadzi do dwóch warunków:

$$\sum_{i=1}^m \alpha_i Y_i = 0, \quad \mathbf{W} = \sum_{i=1}^m \alpha_i Y_i \mathbf{X}_i \quad (5.38)$$

Wektor $\mathbf{W}$, który definiuje hiperpłaszczyznę jest wyrażony przez kombinację wektorów treningowych, a każdy składnik $\mathbf{W}[j]$ jest kombinacją j wartości cech dla wszystkich wektorów $\mathbf{X}_i[j]$.

Zgodnie z warunkami Karush-Kuhn-Thuckera:

$$\alpha_i(Y_i[\mathbf{X}_i \cdot \mathbf{W} + b] - 1) = 0, \quad i = 1, \dots, m \quad (5.39)$$

dla $\alpha_i \neq 0$ wektory muszą leżeć na jednej z hiperpłaszczyzn brzegowych $Y_i[\mathbf{X}_i \cdot \mathbf{W} + b] = 1$; wektory te “wspierają” hiperpłaszczyznę $\mathbf{W}$, która definiuje rozwiązanie problemu optymalizacji. Minimalizacja przebiega najczęściej przy pomocy nieco zmodyfikowanej optymalizacji kwadratowej:

$$\max_{\alpha} \mathbf{W}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j Y_i Y_j \mathbf{X}_i \mathbf{X}_j \quad (5.40)$$

z ograniczeniami:

$$\alpha_i \geq 0 \quad i = 1, \dots, m \quad \sum_{i=1}^m \alpha_i Y_i = 0 \quad (5.41)$$

Funkcja dyskryminująca przyjmuje postać:

$$g(x) = \text{sgn} \left(\sum_{i=1}^m \alpha_i Y_i \mathbf{X} \cdot \mathbf{X}_i + b \right) \quad (5.42)$$

Teraz już dosyć łatwo zastąpić iloczyn skalarny $\mathbf{X} \cdot \mathbf{X}_i$ przez funkcję kernelową $k(\mathbf{X}, \mathbf{X}') = \phi(\mathbf{X}) \cdot \phi(\mathbf{X}')$, gdzie $\phi(\mathbf{X})$ reprezentuje transformację oryginalnych wektorów do nowej przestrzeni kernelowej. Dla dowolnego $\phi(\mathbf{X})$ wektor częściowo prostopadły do przestrzeni objętej przez $\phi(\mathbf{X}_i)$ nie ma wpływu na iloczyn $\phi(\mathbf{X}) \cdot \phi(\mathbf{X}')$, dlatego wystarczy wyrazić $\phi(\mathbf{X})$ i $\mathbf{W}$ jako kombinację wektorów $\phi(\mathbf{X}_i)$. Wymiarowość d wektorów wejściowych jest często niższa od liczby wektorów treningowych $d < m$, wówczas $\phi(\mathbf{X})$ reprezentuje mapowanie do wyższej m -wymiarowej przestrzeni. Podążając za teorią Covera [12] można pokazać, że prawdopodobieństwo liniowej separacji danych wzrasta wraz z wymiarowością przestrzeni, w której te dane się znajdują. W przestrzeni $\phi()$ funkcja dyskryminująca przyjmuje postać:

$$g(\mathbf{X}) = \text{sgn} \left(\sum_{i=1}^m \alpha_i Y_i k(\mathbf{X}, \mathbf{X}_i) + b \right) \quad (5.43)$$

Jeżeli funkcja kernelowa jest liniowa, wówczas przestrzeń $\phi()$ jest zwyczajnie przestrzenią oryginalną, a składowe funkcji dyskryminującej bazują na odległości kosinusowej od wektorów referencyjnych $\mathbf{X}_i$ z klasy Y_i . A zatem oryginalne cechy $\mathbf{X}[j], j = 1..d$ zostają zastąpione przez nowe cechy $Z_i(\mathbf{X}) = k(\mathbf{X}, \mathbf{X}_i)$, które oceniają jak daleko wektor znajduje się od wektorów treningowych. Używając znaków przy współczynnikach wektora $A_i = \alpha_i Y_i$ funkcja dyskryminująca przyjmuje postać:

$$g(\mathbf{X}) = \text{sgn} \left(\sum_{i=1}^m \alpha_i Y_i Z_i(\mathbf{X}) + b \right) = \text{sgn} (\mathbf{A} \cdot \mathbf{Z}(\mathbf{X})) + b \quad (5.44)$$

W przypadku gdy dane nie są separowalne, zamiast odległości lepszym wydaje się być użycie lokalnych miar podobieństwa, przykładowo skalując odległość za pomocą funkcji gaussowskich. Prowadzi to do jednego z najbardziej użytecznych kerneli:

$$k_G(\mathbf{X}, \mathbf{X}') = \exp(-\beta \|\mathbf{X} - \mathbf{X}'\|^2) \quad (5.45)$$

5.7.2 Kernelowe przestrzenie cech

Dla każdego wektora $\mathbf{X}$ można uzyskać nie tylko n cech wejściowych, ale także m cech kernelowych $Z_i(\mathbf{X}) = k(\mathbf{X}, \mathbf{X}_i)$ zdefiniowanych dla każdego wektora treningowego. Biorąc kernel gaussowski $k_G(\mathbf{X}, \mathbf{X}')$ i ustawiając $g(\mathbf{X}) = \text{const}$ otrzymuje się odpowiednik faktu wzięcia ważonej sumy Gaussów scentrowanych wokół pewnych wektorów wsparcia, które znajdują się blisko granicy. Ponieważ kontury funkcji dyskryminujących w przestrzeni kernelowej są w przybliżeniu stałe (podczas gdy $\mathbf{X}$ porusza się wzdłuż nieliniowej granicy decyzji w przestrzeni wejściowej), leżą one na hiperpłaszczyźnie. Zatem w przestrzeni cech kernelowych możliwe jest bezpośrednie użycie dyskryminacji liniowej bez całej maszyneryi SVM-owej. Zostało to zaprezentowane w eksperymentach obliczeniowych poprzez porównanie rezultatów SVM z kernelem gaussowskim (używającym programowania kwadratowego) z bezpośrednim liniowym rozwiązaniem w przestrzeni opartej o cechy kernelowe (por. tabela 5.8) [65].

5.8 Maszyny cech wspierających

W pewnych przypadkach użycie cech kernelowych jest przesadą, albowiem separacja może być osiągnięta przy użyciu oryginalnych cech, które nie są obecne w przestrzeni kernelowej. Załóżmy że dane dla każdej z klas mają rozkład gaussowski (co jest dość częstym przypadkiem), wówczas najlepszy kierunek separujący jest wyznaczony przez różnicę średnich z dwóch klas $\mathbf{W} = \mathbf{m}_1 - \mathbf{m}_2$. Dodając rzut na ten kierunek jako nową cechę $r(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X}$ umożliwiamy znalezienie prostego rozwiązania za pomocą liniowej dyskryminacji.

Maszyny cech wspierających (*Support Features Machines, SFM*), wprowadzone przeze mnie w pracy [65], bazują na generowaniu nowych “cech wspierających” przy pomocy różnych kerneli, przypadkowych liniowych rzutów oraz ograniczonych rzutów, które poprzedzają selekcję cech i liniową dyskryminację. Oczywiście jest fakt, iż możliwe jest użycie także innych algorytmów uczenia maszynowego w tak rozszerzonej przestrzeni cech. W obecnej wersji algorytm używa trzech typów cech, które zostały opisane poniżej.

Cechy pierwszego typu są utworzone przez rzuty na N losowo wygenerowanych kierunków w oryginalnej n -wymiarowej przestrzeni wejściowej. Kierunki te mogą być poprawiane w systematyczny sposób, na przykład przez dodanie kierunków pochodzących z połączenia centrów klastrów zawierających przypadki należące do jednej klasy. Kolejne rzuty na przypadkowe kierunki zwiększają wymiarowość danych, a zgodnie z teorią przedstawioną w pracy [12], im większy jest wymiar przestrzeni cech, tym łatwiejsze staje się odseparowanie danych należących do przeciwnych klas.

Drugi typ cech bazuje na ograniczonych losowych rzutach, tak jak to ma miejsce w algorytmie aRPM (zob. 5.9) [15]. Rzut na przypadkowy kierunek $Z_i(\mathbf{X}) = \mathbf{W}_i \cdot \mathbf{X}$ może nie być zbyt użyteczny, jednakże w pewnych przedziałach wartości Z_i może znajdować się wystarczająco duży, czysty klaster rzutowanych przypadków. Przykładowo, w przypadku problemów parzystości [26, 40] rzuty mają zawsze silnie nakrywające się klasowo-zależne rozkłady prawdopodobieństwa, ale rzuty na kierunki $[1, 1..1]$ pokazują czyste zlokalizowane klastry ze stałą liczbą jedynek (lub zer). Klastry zawierające przypadki treningowe z klasy C mogą być odseparowane od pozostałych przypadków rzutowanych na wymiar Z_i za pomocą funkcji okienkowych $H_i(\mathbf{X}) = H(Z_i(\mathbf{X}); C)$. Powstają zatem cechy binarne $H_i(\mathbf{X}) \in \{0, 1\}$, bazujące na liniowych projekcjach ograniczonych do wycinka przestrzeni prostopadłej do wymiaru Z_i .

Trzeci rodzaj to cechy bazujące na kernelach. Podczas kiedy wiele kerneli może być użytych razem, włączając w to te same kernele ale z różnymi parametrami, w zaprezentowanej tutaj wersji użyte zostały kernele gaussowskie z ustaloną dyspersją β , przyjętą dla każdego wektora treningowego (potencjalnego wektora wsparcia) $K_i(\mathbf{X}) = \exp(-\beta \sum |\mathbf{X}_i - \mathbf{X}|^2)$.

Liczba generowanych cech rośnie liniowo wraz z liczbą przypadków treningowych m , więc dla dużego m powinny one zostać zredukowane przy użyciu prostych filtrów [20]. Przedstawiona tu wersja bazuje na filtrze opartym o współczynnik zysku informacji. Lokalne cechy kernelowe mają wartości bliskie zera za wyjątkiem przypadków znajdujących się w pobliżu wektorów wspierających, dlatego też ich użyteczność powinna być ograniczona do sąsiedztwa $O(\mathbf{X}_i)$, w którym $G_i(\mathbf{X}) > \epsilon$ (w eksperymentach przyjęto $\epsilon = 0.001$). Podobnie dla ograniczonych projekcji, sąsiedztwo ograniczone jest do tych wektorów, które pochodzą z jednej klasy i mieszczą się w interwale $[a, b]$.

Aby zaakceptować nową cechę f spośród cech typu Z, H, K musi ona spełnić trzy warunki:

1. Sąsiedztwo nie powinno być zbyt małe, lokalne cechy powinny zawierać przynajmniej η wektorów;
2. W lokalnym sąsiedztwie $MI(f(\mathbf{X}), C) > \alpha$, zysk informacyjny cechy $f(\mathbf{X})$ nie powinien być zbyt mały;
3. Powinna możliwie najlepiej odseparowywać przypadki należące do różnych klas, a więc jej maksimum prawdopodobieństwa $\max_C p(C|f(\mathbf{X})) > \delta$.

Liczba wektorów w sąsiedztwie η została ustawiona na $\eta = 10$, jednakże w niektórych problemach z bardzo niewielką liczbą wektorów treningowych powinna ona zostać ustawiona na mniejszą wartość. Nieograniczone projekcje zawierają wszystkie dane i nie mogą osiągnąć $p(C|Z(\mathbf{X})) = 1$ dla wszystkich wektorów, dlatego też do ich wyboru został użyty współczynnik zysku informacji. Parametry α i δ zostały ustawione w taki sposób, aby pozostawić wystarczającą liczbę użytecznych cech bazujących na kernelach wspomaganych przez wektory w pobliżu granicy decyzji lub ograniczonych projekcji, które także znajdują się w pobliżu granicy. Parametry te zostały ustawione w taki sposób, aby pozostawić 30% wektorów dla każdego zbioru danych. Ich wpływ na selekcję wektorów wspierających pokazano na rysunku 5.7, gdzie użyto dwóch nakrywających się rozkładów gaussowskich.

Używając parametru β użytkownik może określić poziom rozmycia. Parametr ten może być ustawiony ręcznie lub automatycznie.

Ostatecznie wektor $\mathbf{X}$ jest zatem skomponowany z wielu cech $\mathbf{X} = [X_1 \dots X_n, Z_1 \dots Z_n, H_1 \dots H_n, K_1 \dots K_n]$. W tak skonstruowanej przestrzeni cech algorytm SFM poszukuje liniowego rozwiązania, aczkolwiek warto zaznaczyć że inne metody uczenia mogą znaleźć lepsze rozwiązanie.

Nowe cechy wspierające utworzone w ten sposób bazują na tych transformacjach wejść, które zostały uznane jako interesujące dla określonego problemu i niosą ze sobą jakieś znaczenie i interpretację. Wektory wspierające nie są uczone, ale wybierane z przypadkowych projekcji lub konstruowane przy pomocy zlokalizowanych funkcji kernelowych, a następnie dodawane do puli cech, gdy zawierają interesujące informacje mogące pomóc przy rozwiązaniu problemu. Technicznie rzecz ujmując, oznacza to iż więcej uwagi kładzione jest na generację cech niż na skomplikowaną optymalizację algorytmów czy wymyślanie nowych metod klasyfikacji. Ważność generowania nowych

Schemat 2 Maszyny cech wspierających

Require: Ustaw wartości parametrów α , β , δ i η .

- 1: **for** $i = 0$ to N **do**
 - 2: Losowo wygeneruj nowy kierunek $\mathbf{W}_i \in [0, 1]^n$
 - 3: Dokonaj rzutowania wszystkich $\mathbf{X}$ na ten kierunek $\mathbf{Z}_i = \mathbf{W}_i \cdot \mathbf{X}$ (cechy Z)
 - 4: Przeanalizuj rozkłady $p(Z_i|C)$ aby sprawdzić czy istnieją jakieś czyste klastry,
 - 5: **if** liczba wektorów w klastrze $H_j(Z_i; C)$ przekracza η **then**
 - 6: zaakceptuj nową cechę binarną H_{ij}
 - 7: **end if**
 - 8: **end for**
 - 9: Utwórz cechy kernelowe $K_i(\mathbf{X}), i = 1..m$
 - 10: Zrób ranking wszystkich oryginalnych i dodatkowych cech f_i używając informacji wzajemnej.
 - 11: Usuń cechy, dla których $MI(K_i, C) \leq \alpha$.
 - 12: Usuń cechy, dla których $\max_C p(C|f(\mathbf{X})) < \delta$.
 - 13: Zbuduj liniowy model w rozszerzonej przestrzeni cech.
 - 14: Sklasyfikuj dane testowe zmapowane do rozszerzonej przestrzeni.
-

cech została przeanalizowana w publikacjach [30, 15, 64], aczkolwiek dodawanie cech kernelowych (jak to ma miejsce w SFM) znacznie poprawia rezultaty. W istocie SFM wymaga konstrukcji i selekcji nowych cech, które poprzedzają proste liniowe metody uczenia. Chociaż wiele parametrów może być użytych do kontroli tego procesu, to mogą one zostać ustawione lub wyznaczone w sposób automatyczny. Rozwiązania przy użyciu SFM są bardzo dokładne i proste do zrozumienia.

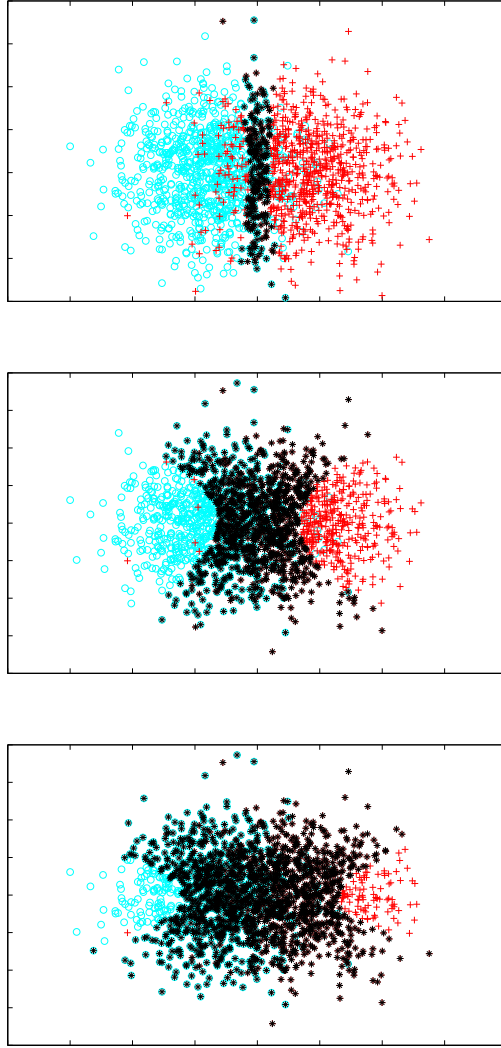
Użyteczność opisywanego algorytmu została przetestowana i porównana z innymi metodami na kilku zbiorach danych. Wyniki (średnie poprawności klasyfikacji i odchylenia standardowe) 10 razy powtórzonych 10-krotnych krosvalidacji umieszczono w tabelach 5.8 i 5.9. Parametry konfiguracyjne wszystkich porównywanych klasyfikatorów były dobierane przy użyciu wewnętrznej krosvalidacji.

Aby sprawdzić wpływ różnych typów cech wsparcia, użyto wszystkich możliwych kombinacji cech. Przyjęto następujące oznaczenia: X to cechy oryginalne, K cechy kernelowe, Z to nieograniczone rzuty na prostą, a H odnosi się do cech opartych na ograniczonych rzutach. Zatem dla każdego zbioru danych utworzono 15 różnych przestrzeni cech: X, K, Z, H, K+Z, K+H, Z+H, K+Z+H, X+K, X+Z, X+H, X+K+Z, X+K+H, X+Z+H, X+K+Z+H, aczkolwiek pokazano tylko te, które oferują najciekawsze rezultaty.

W tabeli 5.8 zestawiono wyniki zoptymalizowanego SVM z kernelem liniowym (SVML) i kernelami gaussowskimi (SVMG), a następnie porównano je z SFM wykonanym na cechach kernelowych.

Dla zbiorów *Ionosphere* i *Sonar* (opis zbiorów w dodatku A) widać dużą korzyść z użycia cech kernelowych zamiast oryginalnych. W przypadku Leukemii prosty model liniowy sprawuje się lepiej, aczkolwiek należy tutaj pamiętać, że zbiór ten zawiera niewielką liczbę wektorów. Przykłady te pokazują dwie rzeczy: po pierwsze, czasami cechy kernelowe są mniej użyteczne od cech oryginalnych, a po drugie, różnice pomiędzy SVMG i SFM(K) wahają się w granicach odchylenia standardowego (zob. tabela 5.8), tak więc jawna reprezentacja kernelowej przestrzeni cech daje takie samo rozwiązanie.

Porównując z wynikami otrzymanymi przez SFM(K) sprawdzono, że dodawanie



Rysunek 5.7: Wpływ parametru α na wybór kerneli użytych jako cechy wspierające. Od góry: $\alpha = 0.005$, $\alpha = 0.05$, $\alpha = 0.1$.

różnych rodzajów cech wsparcia prowadzi do znaczącej poprawy wyników klasyfikacji, aczkolwiek dla różnych zbiorów danych, różne typy cech okazują się być istotne. W przypadku *Appendicitis* ograniczone projekcje prowadzą do znaczącej poprawy poprawności klasyfikacji przy jednoczesnym zmniejszeniu wariancji. Podobnie ma się rzecz ze zbiorami *Heart-c* i *Hepatitis*. Natomiast największą zmianę można zaobserwować na zbiorze *Parity8*, gdzie ograniczone projekcje rozwiązują problem niemal perfekcyjnie. Dla *Ionosphere* wyniki ulegają poprawie gdy doda się cechy kernelowe, zaś w przypadku zbioru *Sonar* najlepsze rezultaty zaobserwować można dla wszystkich typów cech połączonych razem.

Niezaprzeczalnymi zaletami zaprezentowanego powyżej algorytmu jest oferowana przez niego wielorozdzielczość, jak również jawna reprezentacja i systematyczne możliwości rozszerzania przestrzeni cech, podczas gdy wielokernelowe modele są mało wykorzystywane i trudne w użyciu.

Dane	SVML	SVMG	SFM(K)
Appendicitis	87.6±10.3	86.7±9.4	86.8±11.0
Credit-a	85.5±4.3	85.6±6.4	84.2±5.6
Diabetes	76.9±4.5	76.2±6.1	77.6±3.1
Heart-c	82.5±6.4	82.8±5.1	81.2±5.2
Hepatitis	82.7±9.8	82.7±8.4	82.7±6.6
Ionosphere	89.5±3.8	94.6±4.4	94.6±4.5
Leukemia	98.6±4.5	84.6±12.1	87.5±8.1
Parity8	33.4±5.9	12.1±5.9	11±4.3
Sonar	75.5±6.9	86.6±5.8	88.0±6.4

Tabela 5.8: Porównanie SVM i SFM w kernelowej przestrzeni cech.

Dane	K	H	K+H	Z+H	K+H+Z
Appendicitis	86.8±11.0	89.8±7.9	89.8±7.9	89.8±7.9	89.8±7.9
Diabetes	77.6±3.1	76.7±4.3	79.7±4.3	79.2±4.5	77.9±3.3
Heart-c	81.2±5.2	84.8±5.1	80.6±6.8	83.8±6.6	78.9±6.7
Hepatitis	82.7±6.6	83.9±5.3	83.9±5.3	83.9±5.3	83.9±5.3
Ionosphere	94.6±4.5	93.1±6.8	94.6±4.5	93.0±3.4	94.6±4.5
Parity8	11±4.3	99.2±1.6	97.6±2.0	99.2±2.5	96.5±3.4
Sonar	83.6±12.6	66.8±9.2	82.3±5.4	73.1±11	87.5±7.6

Tabela 5.9: SFM w różnych przestrzeniach cech.

5.9 Maszyny prawie przypadkowych projekcji

W biologii prezentacja nowego przypadku aktywuje zwykle dużą liczbę mikroobwodów w korze mózgowej, jednakże konkurencja i lokalne hamowanie pozostawiają tylko niewielką liczbę najbardziej aktywnych obwodów, a więc tych, które zawierają istotną informację. Pojedynczy neuron progowy może następnie odczytać poziom aktywacji określonych obwodów, a następnie określić przynależność prezentowanego przypadku do którejś z kategorii, w oparciu o poziom pobudzeń proporcjonalny do liczby klastrów z każdej kategorii, którą ten przypadek pobudza.

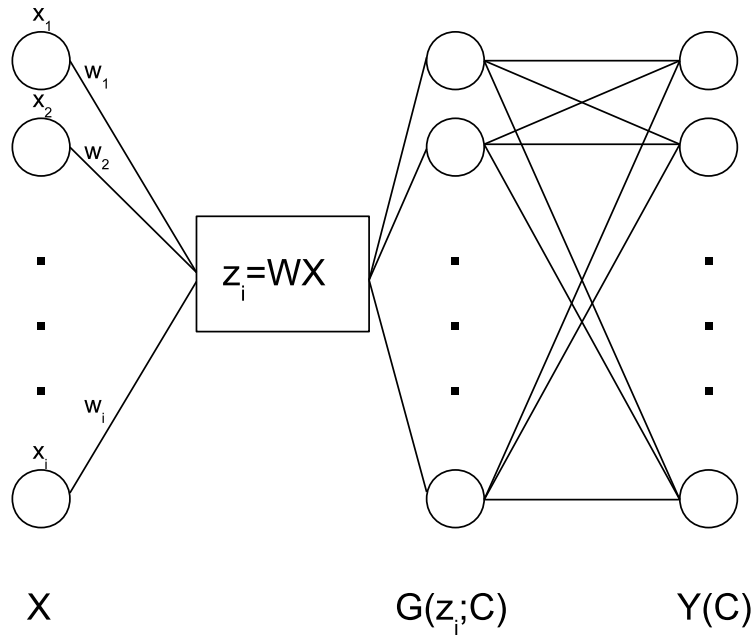
Dwie ważne idee pochodzą z powyższych neuro-kognitywnych inspiracji. Po pierwsze, każdy przypadek powinien zostać przeanalizowany z różnych punktów widzenia, na przykład dzięki utworzeniu interesujących transformacji $\mathcal{T}_i(X)$ obejmujących m.in. rzuty $\mathbf{W}_i \cdot \mathbf{X}$. Mogą one zostać następnie odfiltrowane poprzez zlokalizowane funkcje $\mathcal{T}_i(X) = G_i(\mathbf{W}_i \cdot \mathbf{X})$ odkrywające użyteczne cechy, specyficzne dla danej kategorii. Liczba cech nie powinna być z góry zakładana, lecz dynamicznie zwiększana dopóki nie będzie wystarczającej informacji do podjęcia decyzji. Takie wzajemne oddziaływanie pomiędzy lokalnymi i globalnymi analizami danych wejściowych zostało pominięte zarówno w sieciach neuronowych, jak i innych typach algorytmów uczenia maszynowego.

Transformacje $\mathcal{T}_i(X)$ mapują przypadki wejściowe do jednowymiarowych klastrów, które mogą być albo stosunkowo czyste, albo przynajmniej częściowo dyskryminujące, eliminując pewne kategorie, do których dany przypadek na pewno nie należy [18]. Do dokonania końcowej decyzji powinien zostać użyty, wywodzący się z biologicznych sieci, mechanizm zwycięzca bierze wszystko, bazując na przynależności danego przypadku do rzutowanych klastrów [69].

Schemat 3 Maszyny prawie przypadkowych projekcji

Require: Ustaw dla wszystkich wektorów $\mathbf{X}$ flagę “nowy”.

```
1: for  $i = 0$  to  $N_{rep}$  do
2:   Ustaw przypadkowe wagi  $W_i$ ,  $W_i \in [-1, 1]$ .
3:   Utwórz nowe rzuty  $Z_i = W_i \mathbf{X}$ .
4:   Przeanalizuj  $p(Z_i|C)$  aby wyznaczyć “interesujące” klastry.
5:   Dodaj je jako nowe cechy  $G(Z_i; C)$ , lub opisyane przez klasę ukryte węzły sieci.
6:   Zsumuj aktywność części z ukrytych węzłów dla każdej z klas, aby obliczyć wyjścia
       sieci  $Y(C|\mathbf{X}) = \sum_i G(Z_i; C)$ .
7:   Usuń flagę “nowy” dla wszystkich wektorów, dla których  $Y(C|\mathbf{X}) \geq \beta$ .
8: end for
   Sprawdź sieć.
9: if Dokładność się nie zwiększa then
10:  return sieć.
11: else
12:  goto 1
13: end if
```



Rysunek 5.8: Algorytm aRPM przedstawiony w postaci sztucznej sieci neuronowej.

Algorytm prawie przypadkowych projekcji (*Almost Random Projection Machines*, aRPM) inspirowany przez idee wymienione powyżej, posiada zaledwie kilka parametrów (zob. schemat 5.9). Po pierwsze, aby określić czy rzutowany klastrowy jest interesujący użyto indeksu istotności [20], który bierze pod uwagę tylko “nowe” wektory, a więc te, które nie zostały jeszcze zawarte więcej niż β razy w innych klastrach. Po drugie, aby uzasadnić dodawanie nowych cech, nowo powstałe klastry zwiększające liczbę cech nie powinny być zbyt małe, zawierając przynajmniej α część wszystkich wektorów i przynajmniej jeden nowy wektor. Interwały $G(Z_i; C)$, których zadaniem jest wydobywanie klastrów z projekcji, są wyznaczone poprzez interwały $[\min, \max]$. Liczba powtórzeń N_{rep} została ustawiona na 10.

Wagi w tym algorytmie są wybierane losowo, zaś ich wartości nie zmieniają się. Tak więc nie potrzeba dodatkowego uczenia się wag (w przeciwieństwie do wszystkich innych metod uczenia sieci neuronowych). W niektórych modelach sieci, uczenie ograniczone zostało tylko do liniowej warstwy wyjściowej [32], jednak tutaj zostało ono zastąpione przez proste dodawanie odpowiednich wejść. Najprościej ujmując, aRPM generuje wystarczającą liczbę losowych wag i wybiera najbardziej przydatne, generując użyteczne cechy $G(Z_i; C)$. Każdy węzeł odpowiada zatem perceptronowi obejmującemu klastry, które mogą być otoczone przez przypadki z innych klas. Do podjęcia końcowej decyzji w przedstawionych poniżej przykładach aRPM wykorzystuje mechanizm WTA (zwycięzca bierze wszystko).

Aby porównać aRPM z czterema innymi, popularnymi metodami klasyfikacji, wykonano 10 razy powtórzoną 10-krotną krosvalidację, a uśrednione wyniki (poprawności klasyfikacji i odchylenia standardowe) dla każdego ze zbiorów umieszczono w tabeli 5.10.

Ponadto w każdej z kolumn zanotowano średnią (zaokrągloną do liczby całkowitej) złożoność generowanego modelu: dla C4.5 rozmiar drzewa (ilość liści), dla kNN liczbę najbliższych sąsiadów, dla SVM ilość wektorów wsparcia, a dla MLP i aRPM liczbę ukrytych węzłów. Parametry każdego z użytych klasyfikatorów były automatycznie optymalizowane przy pomocy krosvalidacji.

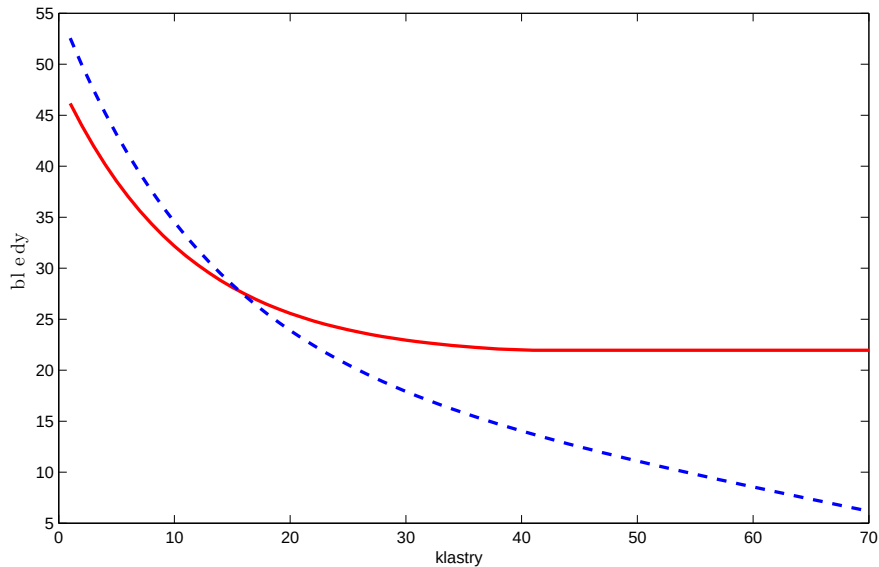
Dane	C4.5	kNN	MLP	SVM	aRPM
Parity8	31.6 ± 1.3 (1)	100 ± 0 (17)	94.1 ± 2.1 (17)	32.4 ± 4.4 (230)	99.2 ± 1.6 (12)
Parity10	40.4 ± 1.6 (1)	100 ± 0 (21)	89.2 ± 12.3 (21)	39.1 ± 6.5 (920)	99.5 ± 0.9 (12)
Breast-w	94.7 ± 2.0 (21)	97.0 ± 1.7 (5)	94.2 ± 0.2 (6)	96.3 ± 2.1 (49)	97.9 ± 1.6 (30)
Heart-c	77.8 ± 2.1 (33)	81.8 ± 6.6 (45)	79.5 ± 1.3 (8)	81.5 ± 1.3 (94)	78.3 ± 4.2 (43)
Leukemia	82.6 ± 8.3 (5)	97.2 ± 1.6 (2)	95.8 ± 3.6 (52)	98.7 ± 3.9 (15)	96.1 ± 8.6 (19)
Liver-disorders	65.8 ± 2.2 (51)	62.0 ± 1.1 (44)	67.5 ± 3.1 (5)	69.2 ± 10.3 (236)	61.1 ± 5.1 (47)

Tabela 5.10: Wyniki poprawności klasyfikacji.

Problem wysokowymiarowej parzystości jest trudny dla większości klasyfikatorów, aczkolwiek algorytm aRPM rozwiązuje go całkiem łatwo w niemal idealny sposób. Warto też zauważyć, że kNN również potrafi sobie poradzić z tym zadaniem, aczkolwiek wymaga do tego $k > 2n$ najbliższych sąsiadów dla n -bitowej parzystości.

Zbiór *Leukemia* (opis zbiorów w dodatku A) został sklasyfikowany przy pomocy aRPM istotnie lepiej, niż zrobiło to drzewo C4.5 i na podobnym poziomie jak reszta porównywanych metod. Dla *Heart-c* opisywany algorytm daje dokładność $78 \pm 4\%$, co nie jest zbyt mocno różne od innych metod, ponieważ wariancja jest dosyć duża. SVM dla tego przypadku radzi sobie całkiem dobrze, aczkolwiek należy zauważyć, że używa on prawie 100 wektorów wsparcia, co oznacza że budowany przez niego model jest raczej skomplikowany. Zbiór *Breast-w* jest klasyfikowany przez aRPM z większą dokładnością, niż czynią to pozostałe metody, zaś porównując wyniki dla *Liver-disorders*, widać że wszystkie rezultaty nie są statystycznie istotnie różne, przez wzgląd na duże wariancje.

Warto się zastanowić, ile powinno zostać utworzonych ukrytych węzłów w algorytmie aRPM. Jeżeli parametr α będzie mały, wówczas będzie ich tworzonych więcej, powodując wzrost dokładności na zbiorze treningowym, jednakże klastry jakie będą się tworzyć będą małe, a generalizacja niewielka. Typowa zbieżność w zależności od liczby węzłów jest monotoniczna i dosyć szybka (patrz rys. 5.9). Dla prezentowanego przykładu (zbiór *Heart-c*) optymalna liczba ukrytych węzłów wynosi ≈ 43 dając 78% poprawności klasyfikacji na zbiorze testowym.

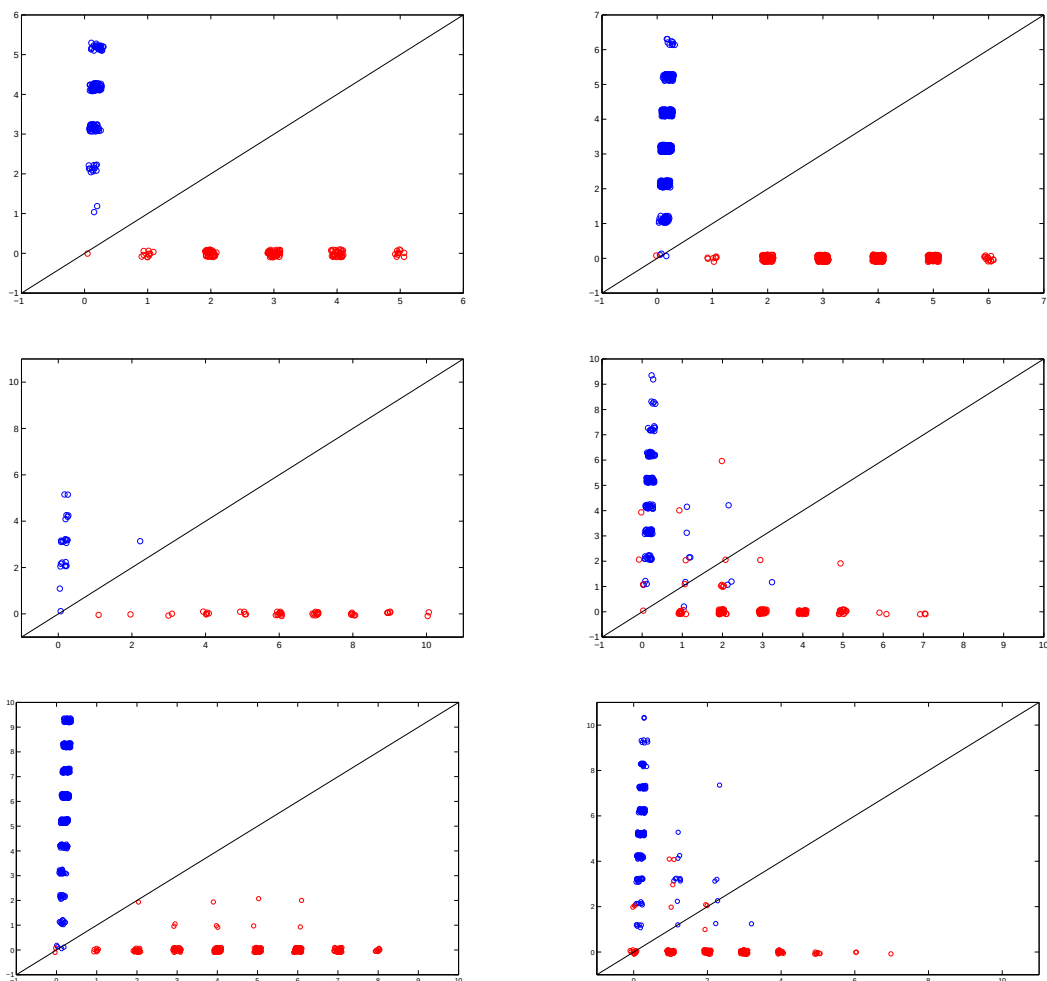


Rysunek 5.9: Wykres pokazujący zależność liczby błędów od ilości węzłów (zbiór *Heart-c*); linia przerywana niebieska - błędy treningowe, linia ciągła czerwona - błędy testowe.

Algorytm aRPM umożliwia łatwą ocenę zaufania dotyczącą otrzymywanych rezultatów (można to zobaczyć na rysunku 5.10). Sieć posiada dwa liniowe wyjścia (dla przypadków dwuklasowych), a ich wartości równe są liczbie aktywowanych węzłów w warstwie ukrytej dla każdej z klas. Każdy z rysunków pokazuje wyjście modelu aRPM, nauczonego na części treningowej wewnątrz krosvalidacji, a następnie użytego do wizualizacji całości danych (pokazując błędy treningowe i testowe). Większość wektorów aktywuje tylko węzły z poprawnej klasy, część przypadków nie trafia do żadnego z klastrów i wpada do klastra (0,0), wówczas mogą być one przypisane do klasy dominującej.

Informatyka neurokognitywna [22] czerpie inspiracje z neurobiologicznych procesów odpowiedzialnych za uczenie. Jednak niewiele z nich zostało użytych w inteligencji obliczeniowej. Pomimo faktu iż nasza współczesna wiedza o tym jak działa mózg jest dosyć ograniczona, nie należy się wzorować tylko na perceptronach, wiele więcej inspiracji może zostać użytych w praktycznym uczeniu i algorytmach rozpoznawania wzorców. W niniejszym rozdziale przedstawiono argumenty, które sugerują, iż bardziej uzasadnione biologicznie wydają się być próby wykorzystania przypadkowych projekcji, niż powolnych mechanizmów uczenia opartych na korekcji błędów. Zaprezentowano tylko najprostszą wersję algorytmu aRPM. Zaskakującym jest, że wyniki modelu, który nie przeprowadza żadnej optymalizacji są dla popularnych zbiorów danych co najmniej tak dobre, jak te uzyskane przez sieci MLP. Dodatkowo metoda ta idealnie radzi sobie z problemami o złożonej logice, takimi jak parzystość w wysoko-wymiarowej przestrzeni, które okazują się być ciężkim wyzwaniem dla wielu popularnych metod uczenia maszynowego (np. SVM).

Mózg nie używa ustalonej liczby cech tak jak robi to większość algorytmów rozpoznawania wzorców, ale zaczyna od małej liczby cech, aktywnie poszukując nowych, najbardziej dyskryminujących, dostarczonych przez neuronowe filtry. Każdy obiekt jest rozpoznawany przy użyciu różnych cech, a mianowicie takich, które najlepiej go



Rysunek 5.10: Wyjście algorytmu aRPM dla danych treningowych i testowych; górny wiersz: *Parity8* i *Parity10*, środkowy wiersz: *Leukemia* i *Heart-c*, dolny wiersz: *Breast-w* i *Liver-disorders*.

charakteryzują. Zatem można powiedzieć, że selekcja i konstruowanie cech nie jest żadnym oddzielnym procesem, lecz składową procesy uczenia i kategoryzacji. Dodatkowo opisywany tutaj algorytm umożliwia wizualizację wyjść (patrz rys. 5.10), oraz potrafi oszacować poziom zaufania dokonanej klasyfikacji pokazując, jak daleko od granicy jest dany przypadek. Jest to cecha niezwykle użyteczna np. w problemach medycznych.

Finalnym celem uczenia jest kategoryzacja, ale pośrednie etapy są także ważne. Znajdowanie interesujących spojrzeń na dane lub konstrukcja filtrów interesujących informacji są bardzo istotne. Każdy filtr dokonuje swojej własnej selekcji lub ważenia cech. W przeciwieństwie do używania sieci z ustaloną liczbą wejść, systemy, które aktywnie badają dane, próbując “zobaczyć je” poprzez swoje filtry, okazują się być niezwykle potrzebne. Kiedy otrzymają one odpowiednią ilość informacji do sklasyfikowania danych, wówczas ich praca zostaje zakończona. Otwiera to drogę do nowych algorytmów, zdolnych do uczenia się z przypadków, które mają rozmaite struktury, włączając w to takie, które cechują się wieloma brakującymi wartościami.

5.10 Maszyny prawie przypadkowych projekcji z maksymalizacją marginesu

W tej wersji (opisywanego w poprzednim podrozdziale) algorytmu aRPM największą uwagę kładzie się na maksymalizację marginesu klasyfikacji. Nowe projekcje powinny być dodawane tylko wtedy, jeżeli zwiększają poprawność klasyfikacji tych przypadków, które znajdują się albo po złej stronie, albo w pobliżu granicy decyzyjnej.

Algorytm aRPM może przekształcać przestrzeń cech wejściowych $\mathcal{X}$ w przestrzeń cech wsparcia $\mathcal{H}$ przy pomocy wielu rozmaitych, użytecznych cech. Końcowa analiza w przestrzeni $\mathcal{H}$ może zostać zrealizowana przez dowolny algorytm klasyfikacyjny, jednakże tutaj używane są dwa rozwiązania - głosowanie większościowe i liniowa dyskryminacja.

W tej wersji algorytmu używane są tylko dwa rodzaje cech - cechy wygenerowane przy pomocy rzutów na losowe kierunki, oraz cechy oparte o gaussowskie kernele. Cechy te są następnie odfiltrowywane przez indeks oceniający szerokość marginesu klasyfikacji.

Tak jak w poprzedniej wersji algorytmu, tak i tutaj po rzutowaniu danych na losowe kierunki, poszukiwane są przedziały $[T_a, T_b]$ (klastry), w których występują wektory tylko z jednej klasy. Prowadzi to do utworzenia cech binarnych $B_i(\mathbf{X}) = T(\mathbf{X}; \mathbf{W}_i, [T_a, T_b]) \in \{0, 1\}$, opartych na liniowym rzucie na kierunek $\mathbf{W}_i$.

Dobra potencjalna cecha powinna zawierać pewną minimalną liczbę η wektorów treningowych. Optymalna liczba będzie zależeć od danych, dlatego też jest ona wyznaczana w procesie krosvalidacji z przedziału 4-15. Kiedy nowa cecha zostanie zaakceptowana, poszukiwanie nowych kierunków jest kontynuowane kilka razy (w tej wersji liczba ta została ustawiona na $N_{rep} = 20$) i jeżeli nie zostaną znalezione żadne użyteczne cechy, procedura zostaje zakończona i przechodzi do generowania innych typów cech.

Drugi typ cech tu uwzględniany to atrybuty oparte na gaussowskich kernelach. W tej wersji algorytmu używanych jest kilka wartości parametru σ w celu utworzenia potencjalnych wektorów wsparcia $G(\mathbf{X}; \mathbf{X}_i, \sigma) = \exp(-\|\mathbf{X}_i - \mathbf{X}\|^2/2\sigma^2)$. Lokalne cechy kernelowe mają wartości bliskie zero, poza najbliższym otoczeniem wektorów wsparcia $\mathbf{X}_i$. Tak więc ich użyteczność jest ograniczona do sąsiedztwa $O(\mathbf{X}_i)$, w którym $G_i(\mathbf{X}) > \epsilon$. W tym sąsiedztwie powinno się znaleźć co najmniej η wektorów, inaczej taka cecha zostanie usunięta. Aby zapewnić wielorozdzielczość, najpierw tworzone są cechy z dużą wartością parametru σ , oferujące gładkie granice decyzji, następnie używana jest znacznie mniejsza wartość, aby utworzyć cechy bardziej zlokalizowane. Do wygenerowania cech kernelowych użyto 5 wartości dyspersji: $\sigma \in \{2^5; 2^2; 2^{-1}; 2^{-4}; 2^{-7}\}$. Potencjalna cecha jest używana na stałe tylko wówczas, jeżeli zwiększa margines klasyfikacji.

Algorytm inkrementacji rozszerza przestrzeń cech aż do momentu, kiedy nie uzyskuje się wzrostu dokładności klasyfikacji. Zwiększanie przestrzeni cech powinno odsuwać wektory dalej od granicy decyzji. Jednakże przesuwac się powinny (w odpowiednim kierunku) tylko te wektory, które są po złej stronie lub znajdują się blisko granicy.

Końcowa decyzja jest wyznaczana przy pomocy mechanizmu WTA, sumując aktywność ukrytych węzłów. Rzuty z dodanymi przedziałami dają binarne aktywacje $B_i(\mathbf{X})$, ale wartości cech kernelowych $G(\mathbf{X}; \mathbf{X}_i, \sigma)$ powinny zostać zsumowane, dając całkowitą aktywację $A(C|\mathbf{X})$ dla każdej klasy. Dodatkowo można określić prawdopodobieństwo klasyfikacji $p(C|\mathbf{X})$ przez podzielenie tych wartości przez całkowitą sumę aktywacji dla wszystkich klas.

Wykreślając $A(C|\mathbf{X})$ kontra $A(\neg C|\mathbf{X})$ dla każdego wektora otrzymuje się

skaterogramy, które można zobaczyć na rysunkach 5.11 i 5.12. Pokazują one jak daleko dany wektor znajduje się od granicy decyzji.

W mechanizmie WTA różnica $|A(C|\mathbf{X}) - A(-C|\mathbf{X})|$ określa odległość od granicy decyzji.

Określenie poziomu zaufania modelu do wektora $\mathbf{X} \in C$ przy pomocy funkcji logistycznej: $F(\mathbf{X}) = 1/(1 + \exp(-(A(C|\mathbf{X}) - A(-C|\mathbf{X}))))$ daje wartości około 1, jeżeli $\mathbf{X}$ jest po dobrej stronie i daleko od granicy, oraz zmierza do zera, jeżeli znajduje się po złej stronie.

Całkowity poziom zaufania modelu może być wyznaczony przez sumę po wszystkich wektorach i powinien wynosić n dla doskonałej separacji danych. Końcowy wpływ dodania nowej cechy $H(\mathbf{X})$ na całkowity poziom zaufania jest równy:

$$U(\mathcal{H}, H) = \sum_{\mathbf{X}} (F(\mathbf{X}; \mathcal{H} + H) - F(\mathbf{X}; \mathcal{H}))$$

Jeżeli $U(\mathcal{H}, H) > \alpha$, wówczas nowa cecha zostaje zaakceptowana, przyczyniając się do zwiększenia marginesu. Parametr α został ustawiony na 0.01, a jego niższe wartości prowadzą do szybszego powiększania się przestrzeni cech dodając cechy o słabszej użyteczności.

Do określenia ostatecznej decyzji algorytm aRPM z maksymalizacją marginesu używa mechanizmu WTA albo liniowej dyskryminacji. Podsumowanie algorytmu pokazano na schemacie 4.

Początkowa przestrzeń $\mathcal{H}$ jest tworzona przy pomocy pojedynczych cech X_i . W ten sposób zachowane zostają także oryginalne cechy, które mogą się okazać bardzo użyteczne.

Do porównania metody aRPM (z i bez maksymalizacji marginesu) z innymi popularnymi klasyfikatorami przeprowadzono 10 razy powtórzoną 10-krotną krosvalidację, a wyniki w postaci uśrednionych dokładności klasyfikacji i odchyłeń standardowych przedstawiono w tabeli 5.11. Przy porównaniu użyto rezultatów otrzymanych z algorytmów Naiwnego Bayesa, kNN (z optymalną liczbą sąsiadów k i odległością Euklidesową), drzewa decyzji SSV [39] oraz SVM z liniowym i gaussowskim kernelem (z optymalnymi wartościami parametrów C i σ dobranymi w krosvalidacji).

W kolumnie aRPM-no przedstawiono rezultaty algorytmu aRPM bez maksymalizacji marginesu, zaś kolejne dwie kolumny zawierają wyniki opisywanej tutaj metody z maksymalizacją marginesu, używającej mechanizmu WTA i liniowej dyskryminacji.

W każdym przypadku dodanie optymalizacji marginesu przyczynia się do poprawy rezultatów. Warto zauważyć, że użycie liniowej dyskryminacji na rozszerzonej przestrzeni cech oferuje najlepsze (lub istotnie nie gorsze) rezultaty. W większości przypadków otrzymane wyniki są lepsze od liniowego i gaussowego SVMa. Jedynymi używanymi tutaj do porównania nieseparowalnymi problemami są parzystość 8 i 10 bitowa, które zostały niemal idealnie sklasyfikowane przy użyciu metody aRPM połączonej z WTA.

Efekt maksymalizacji marginesu jest w oczywisty sposób widoczny na skaterogramach (rysunki 5.11 i 5.12). Większość wektorów, które nie zostały sklasyfikowane z wysokim poziomem zaufania, została odsunięta od granicy decyzji.

Algorytm aRPM [15] został usprawniony na dwa sposoby, przez wybieranie takich węzłów sieci, które zapewniają szeroki margines oraz przez dodanie cech kernelowych i cech oryginalnych. Rezultaty na problemach benchmarkowych pokazują przewagę algorytmu aRPM-L nad resztą klasyfikatorów dla większości przypadków oraz przewagę aRPM-W na problemach parzystości, co świadczy o tym, że metoda ta będzie sobie

Schemat 4 aRPM z maksymalizacją marginesu

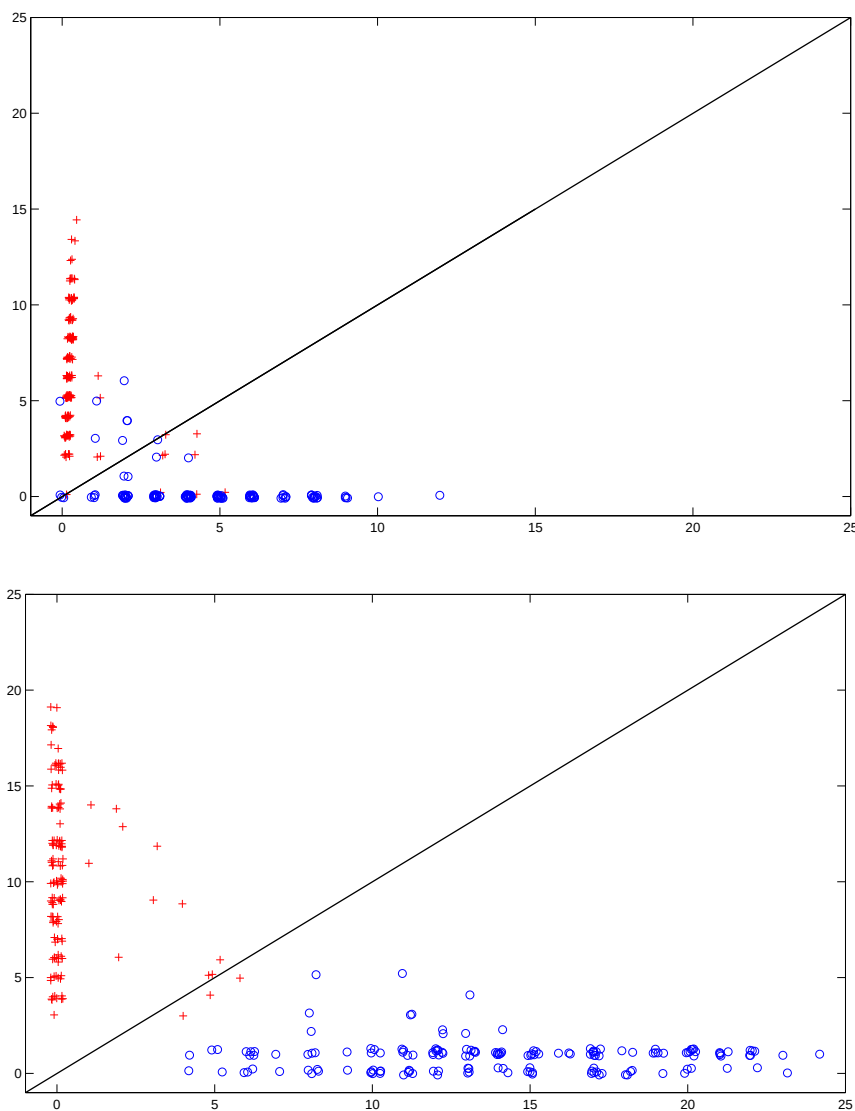
Require: Ustaw wartości parametrów η i α oraz zestaw wartości dyspersji σ (posortowanych malejąco).

- 1: Zestandaryzuj dane zawierające n wektorów i d cech.
 - 2: Ustaw początkową przestrzeń $\mathcal{H}$ używając cech wejściowych x_i .
 - 3: **for** $k = 0$ **to** N_{rep} **do**
 - 4: Wygeneruj losowy kierunek $\mathbf{W} \in [0, 1]^d$
 - 5: Utwórz nowe rzuty $T(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X}$.
 - 6: Posortuj wartości $T(\mathbf{X})$ w kolejności rosnącej, wraz z przypisanymi do nich etykietami klas.
 - 7: Przeanalizuj rozkłady $p(T|C)$ aby znaleźć wszystkie przedziały tworzące czyste klastry, wyznaczając cechy binarne $B_i(\mathbf{X}; C)$.
 - 8: **if** liczba wektorów objętych przez cechę $B_i(\mathbf{X}; C) > \eta$ and $U(\mathcal{H}, B_i) > \alpha$ **then**
 - 9: Zaakceptuj opisaną przez klasę nową cechę binarną $B_i(\mathbf{X})$.
 - 10: **goto** 3
 - 11: **end if**
 - 12: **end for**
 - 13: **for** $j = 1$ **to** n_σ **do**
 - 14: Ustaw wartość dyspersji $\sigma = \sigma_j$.
 - 15: Utwórz cechy kernelowe $G_i(\mathbf{X}) = \exp(-||\mathbf{X}_i - \mathbf{X}||^2/2\sigma^2)$.
 - 16: **if** $U(\mathcal{H}, G_i) > \alpha$ **then**
 - 17: Zaakceptuj opisaną przez klasę nową cechę kernelową $G_i(\mathbf{X})$.
 - 18: **end if**
 - 19: **end for**
 - 20: Zsumuj aktywność ukrytych węzłów dla każdej z klas aby obliczyć wyjścia sieci.
 - 21: Sklasyfikuj dane testowe zmapowane do rozszerzonej przestrzeni cech.
-

Dane	NB	kNN	SSV	SVM(L)	SVM(G)	aRPM-no	aRPM-W	aRPM-L
Appendicitis	83.1 ± 10.2	87.0 ± 10.6	87.9 ± 7.4	85.1 ± 6.0	85.9 ± 6.4	82.6 ± 9.3	87.7 ± 8.1	88.0 ± 6.7
Diabetes	68.1 ± 2.3	75.2 ± 4.1	73.7 ± 3.8	76.4 ± 4.7	75.7 ± 5.9	67.7 ± 4.2	61.2 ± 5.7	76.7 ± 4.4
Glass	68.6 ± 9.0	69.7 ± 7.4	69.7 ± 9.4	40.2 ± 9.6	63.2 ± 7.7	65.0 ± 9.9	60.3 ± 8.5	68.9 ± 8.3
Heart-c	76.5 ± 8.6	82.8 ± 6.7	74.7 ± 8.7	83.2 ± 6.2	83.5 ± 5.3	78.3 ± 4.2	80.1 ± 7.5	83.1 ± 4.7
Liver-disorders	58.6 ± 3.8	62.6 ± 8.5	68.9 ± 9.7	68.4 ± 5.9	69.0 ± 8.4	61.1 ± 5.1	67.5 ± 5.5	72.7 ± 7.9
Wine	98.3 ± 2.6	94.9 ± 4.1	89.4 ± 8.8	96.0 ± 5.9	97.8 ± 3.9	68.6 ± 7.8	94.3 ± 5.8	97.7 ± 4.0
Parity8	28.9 ± 4.6	100 ± 0	49.2 ± 1.0	34.1 ± 11.7	15.6 ± 22.7	99.2 ± 1.6	100 ± 0	34.7 ± 3.8
Parity10	38.1 ± 3.3	100 ± 0	49.8 ± 0.3	44.1 ± 5.0	45.6 ± 4.3	99.5 ± 0.9	100 ± 0	40.3 ± 2.7

Tabela 5.11: Porównanie wyników poprawności klasyfikacji kilku popularnych algorytmów z metodami aRPM-no (wersja bez maksymalizacji marginesu), aRPM-W (wersja z maksymalizacją marginesu i mechanizmem WTA) oraz aRPM-L (wersja z maksymalizacją marginesu i LDA).

świetnie radzić w rozwiązywaniu trudnych problemów ze złożoną logiką. Skaterogramy pokazują efekt optymalizacji marginesu i umożliwiają estymację poziomu zaufania klasyfikacji określonego zbioru danych.

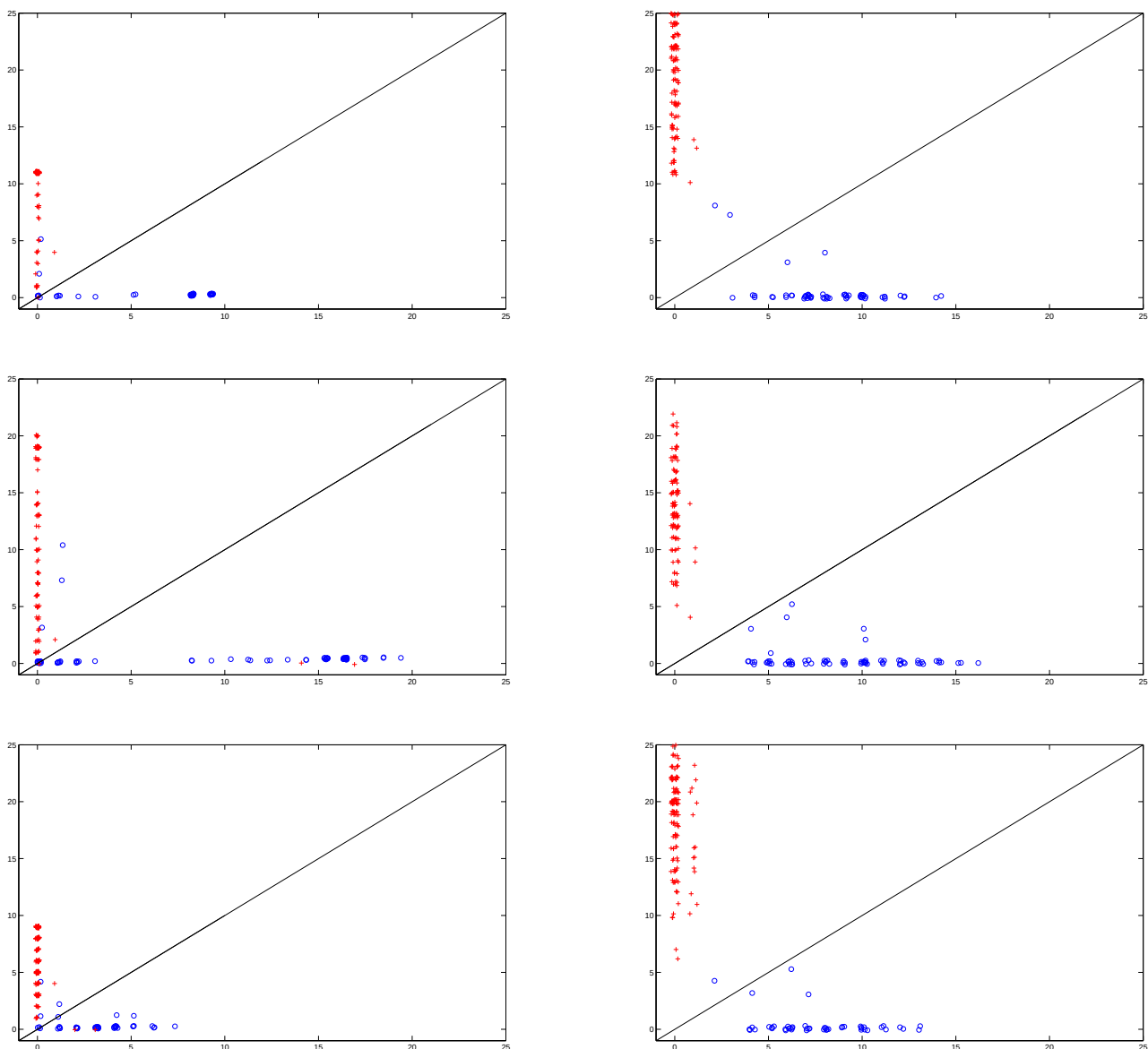


Rysunek 5.11: Wyjście algorytmu aRPM dla zbioru *Heart-c* bez i z optymalizacją marginesu.

5.11 Wygląd granic decyzji dla różnych klasyfikatorów

Z geometrycznego punktu widzenia, każdemu wektorowi treningowemu $\mathbf{X}$ można przypisać punkt w odpowiednio określonej przestrzeni cech, której wymiar jest równy wymiarowi wektora treningowego. Klasyfikator dzieli przestrzeń cech na obszary decyzyjne w taki sposób, aby wszystkim punktom znajdującym się w danym obszarze odpowiadała taka sama decyzja (przypisywana zostawała ta sama klasa). Granice pomiędzy tymi obszarami nazywane są powierzchniami decyzyjnymi lub granicami decyzji. Kształty obszarów decyzyjnych zależą zarówno od danych treningowych, jak i od użytego klasyfikatora.

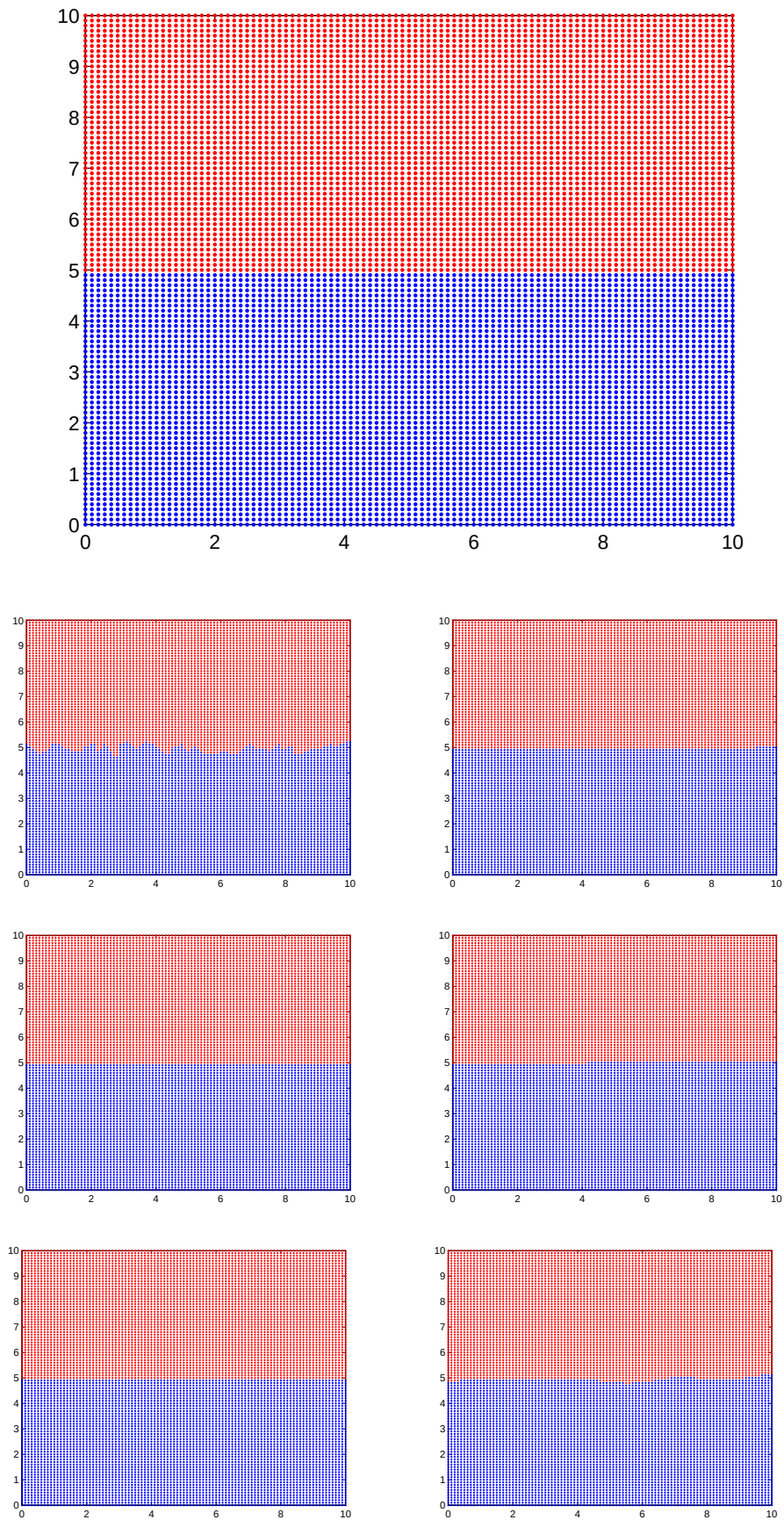
Poniżej przedstawiono (dla kilku różnych typów danych) jak kształtują się granice decyzji dla różnych algorytmów (rysunki 5.13-5.19). Wygenerowano je w taki sposób, aby pokazać zarówno generalizację jak i kształt granicy. Z każdego z zaprezentowanych zbiorów (składającego się z 10000 przypadków) wylosowano 10% wektorów, następnie zbudowano na takim podzbiórze model za pomocą jednego z klasyfikatorów i przetestowano go na pełnym zbiorze danych. Dla metod SVML i SVMG przyjęto stałe wartości parametrów, wynoszące $C = 1$ i $\sigma = 1$.



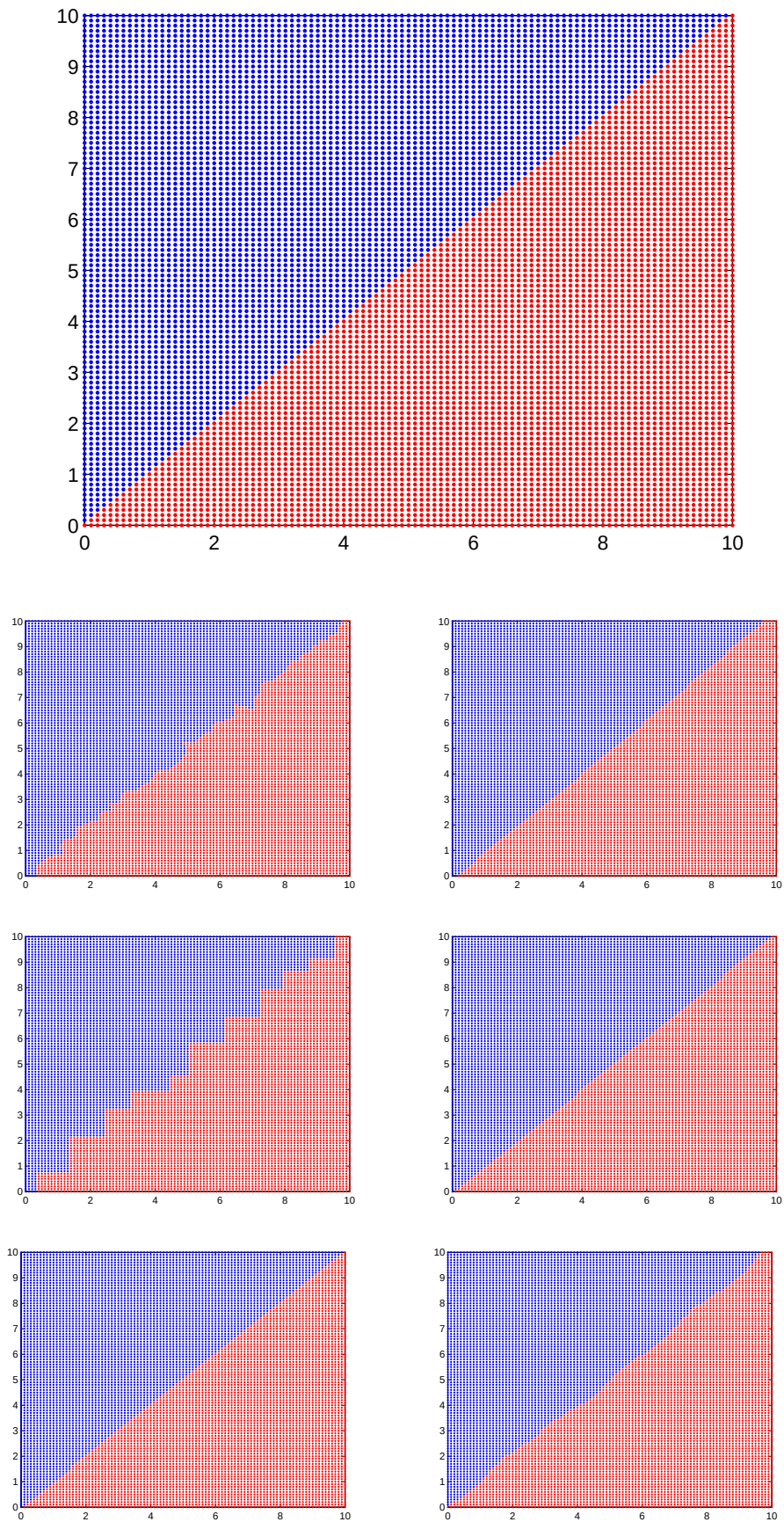
Rysunek 5.12: Wyjście algorytmu aRPM dla zbioru *Wine* bez i z optymalizacją marginesu. Każdy wiersz przedstawia inną klasę (jedna klasa kontra reszta).

Widać, że w przypadku, gdy potrzebne są proste granice decyzji, w zupełności wystarczy model liniowy, gdy granice są ostre najlepiej sprawuje się drzewo decyzji, natomiast kiedy granica jest mocno nieregularna i skomplikowana, optymalnym wyborem wydaje się być klasyfikator oparty o metody kernelowe.

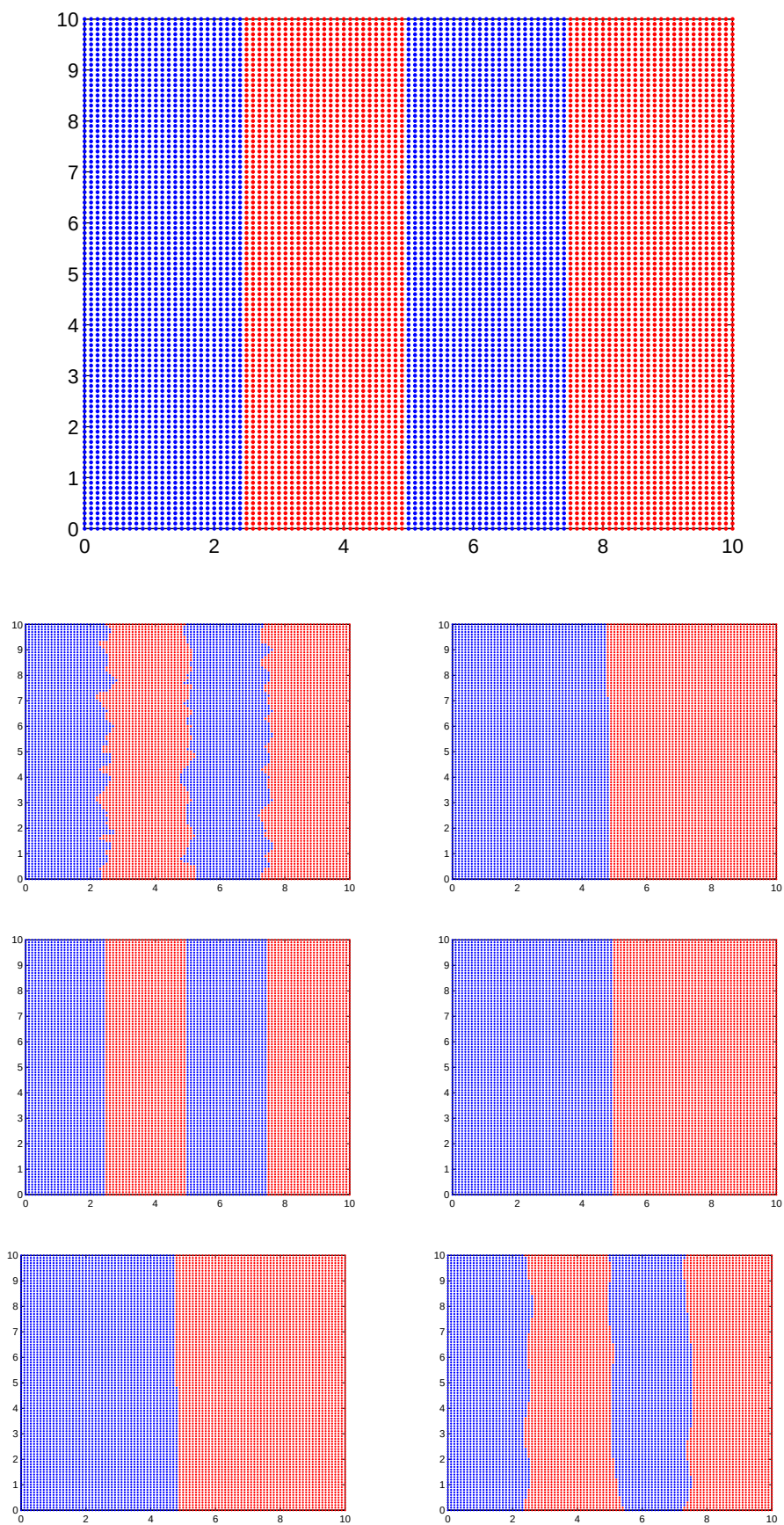
Dodatkowo na rysunku 5.20 pokazano wpływ zmiany parametru σ w SVMG na kształt granicy decyzji. Widać wyraźnie, że im mniejsza jego wartość, tym bardziej zlokalizowane Gaussy i co za tym idzie większa możliwość przeuczenia się algorytmu. Zwiększając jego wartość do $\sigma = 2^3$ otrzymuje się granicę decyzji zbliżoną do tej oferowanej przez SVML (por. rysunek 5.16).



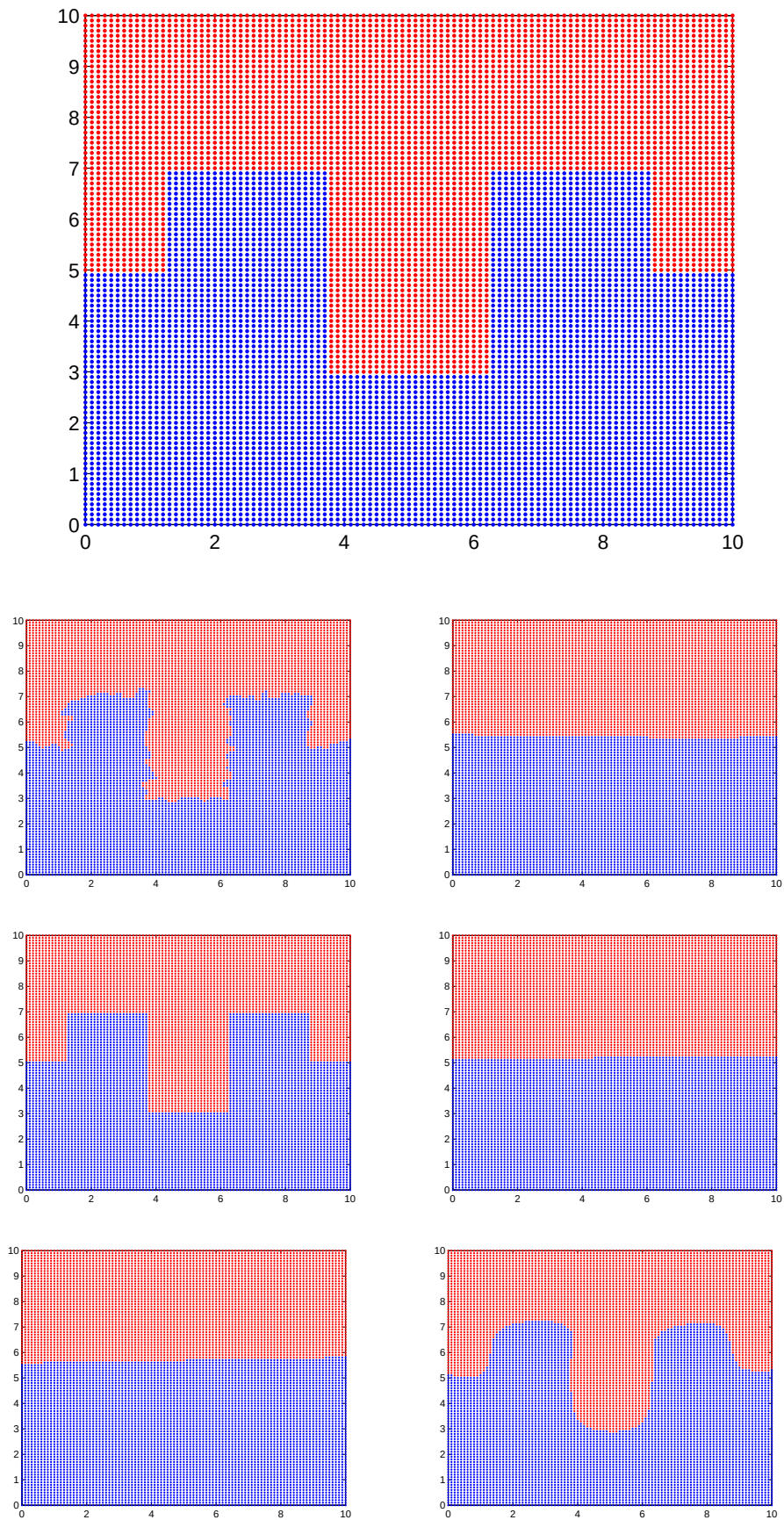
Rysunek 5.13: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVML i SVMG.



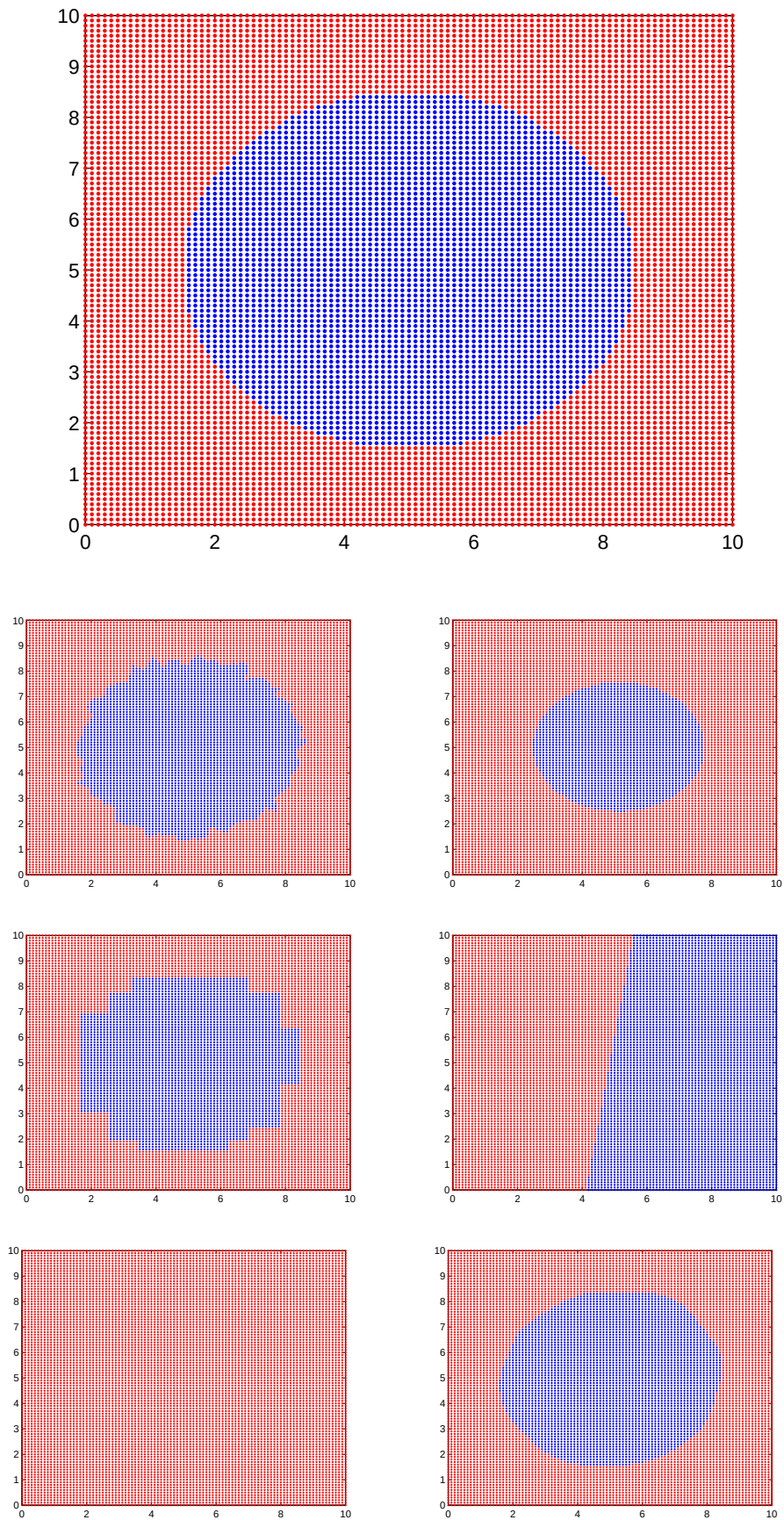
Rysunek 5.14: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVML i SVMG.



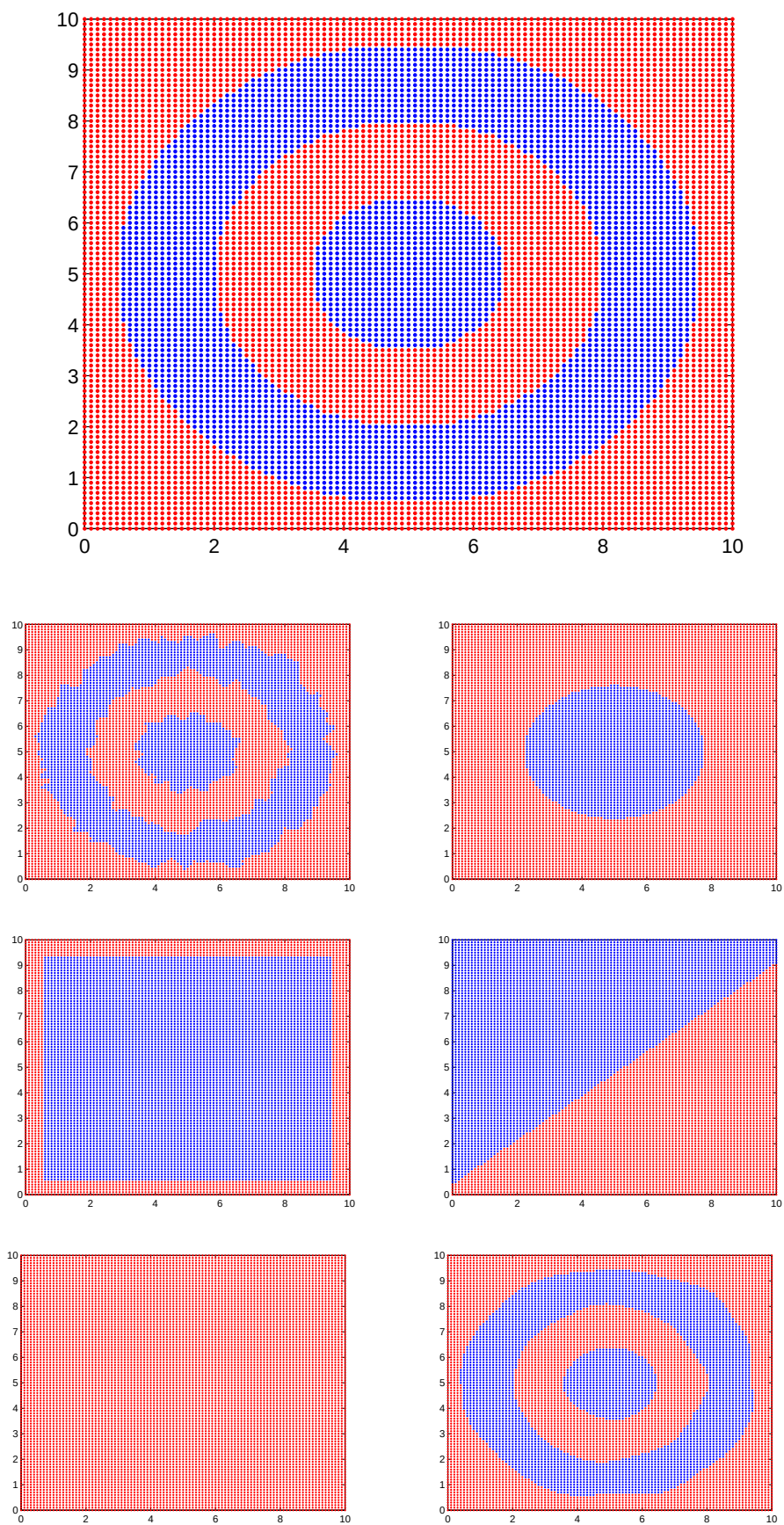
Rysunek 5.15: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVM i SVMG.



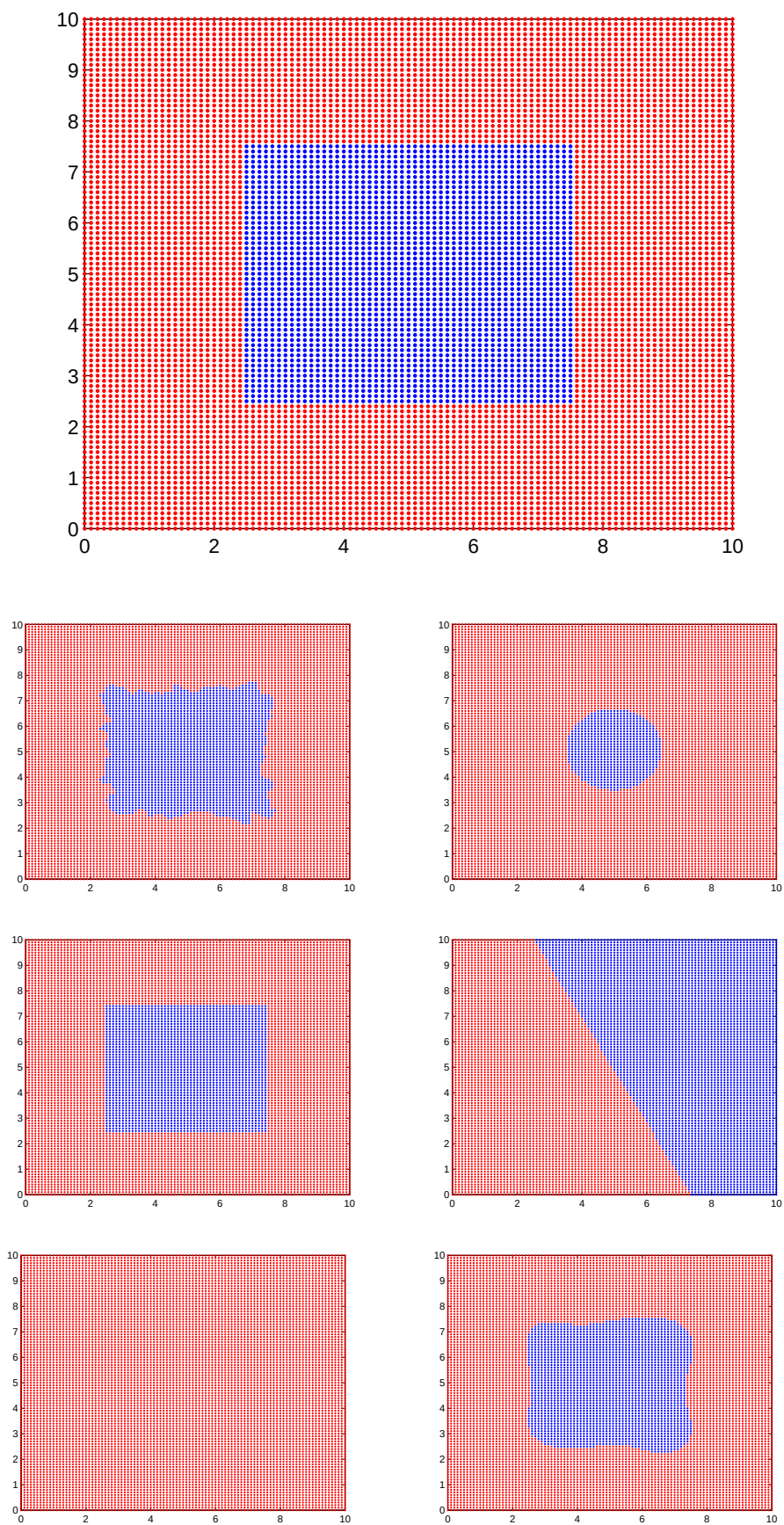
Rysunek 5.16: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVM i SVMG.



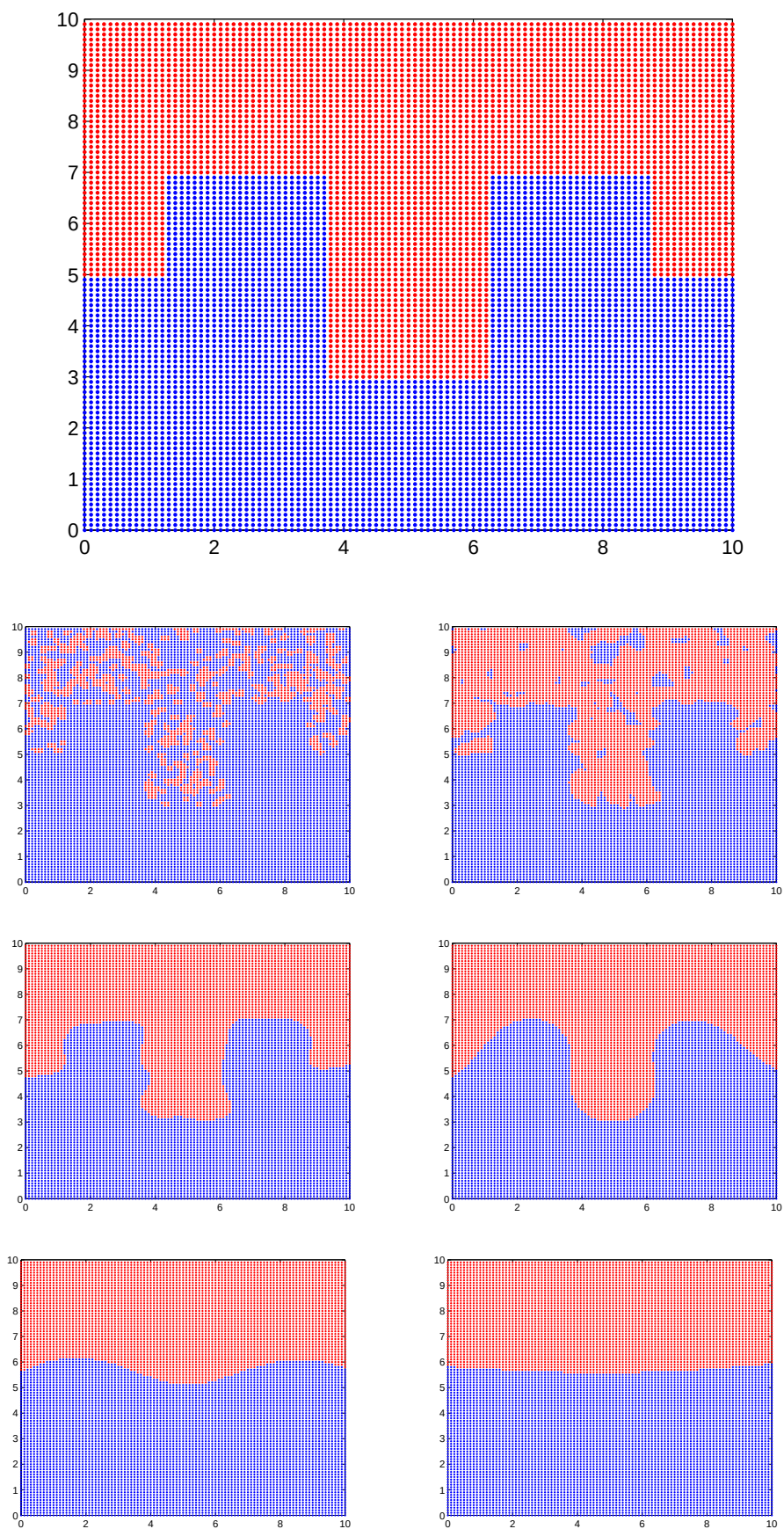
Rysunek 5.17: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVM i SVMG.



Rysunek 5.18: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVML i SVMG.



Rysunek 5.19: Wygląd granic decyzji dla różnych klasyfikatorów; górny wiersz - obraz danych, drugi wiersz - 1NN i NB, trzeci wiersz - C45 i LDA, dolny wiersz - SVML i SVMG.



Rysunek 5.20: Wygląd granic decyzji SVMG dla różnych wartości parametru σ ; górny wiersz - obraz danych, drugi wiersz - $\sigma = 2^{-4}$ i $\sigma = 2^{-3}$, trzeci wiersz - $\sigma = 2^{-1}$ i $\sigma = 2^1$, dolny wiersz - $\sigma = 2^2$ i $\sigma = 2^3$.

Rozdział 6

Konstrukcja nowych, użytecznych cech

W poniższym rozdziale przeanalizowano różne rodzaje przekształceń generujących nowe, użyteczne cechy a także omówione zostały metody ich generowania. Należą do nich: ograniczone losowe projekcje, optymalizacja tych cech przy użyciu metod poszukiwania projekcji, cechy oparte na podobieństwie, cechy kernelowe, cechy pochodzące z częściowo nauczonych różnych algorytmów uczenia się oraz wykorzystanie całych modeli uczenia się jako nowych cech.

Celem uczenia w rozszerzonej przestrzeni cech jest stworzenie takiego obrazu danych wejściowych, który może być obsługiwany przez stosunkowo proste procesy decyzyjne. Skoncentrowano się na hierarchicznej metodzie generowania informacji, począwszy od nowych, prostych cech, poprzez cechy bardziej złożone, a kończąc na cechach opartych o wydobywanie informacji z nauczonych modeli. Przedyskutowano także neurokognitywne inspiracje, które odnoszą się do wydobywania użytecznych informacji z danych oraz uczenia się od innych.

6.1 Neurokognitywne inspiracje

Mózg w dalszym ciągu lepiej radzi sobie niż algorytmy statystyczne lub uczenie maszynowe w rozwiązywaniu wielu złożonych problemów, które wymagają analizy sygnałów. Nasuwa się pytanie - dlaczego mózg radzi sobie tak dobrze z tak skomplikowanymi zadaniami, a uczenie maszynowe jest tak słabe, mimo rozwoju zaawansowanych algorytmów statystycznych, sieci neuronowych i innych biologicznie inspirowanych metod inteligencji obliczeniowej?

Sztuczne sieci neuronowe czerpią inspirację z neuronowego przetwarzania informacji na poziomie pojedynczego neuronu. Próby zrozumienia mikroukładów i czerpania inspiracji z funkcji całych kolumn kory nowej do tej pory nie były zbyt udane. Projekt The Blue Brain [59] zapewniał biologicznie dokładną symulację kolumn kory nowej, ale nie dostarczył on żadnych ogólnych zasad, na jakich te kolumny działają. Neuronauki obliczeniowe (*computational neuroscience*) odgrywa bardzo ważną rolę w zrozumieniu szczegółów jak działają i jakie funkcje pełnią neurony, ale może nie być najkrótszą drogą do znacznego postępu w inteligencji obliczeniowej.

Neurokognitywne inspiracje, w oparciu o ogólne zrozumienie funkcji mózgu mogą być bardzo przydatne przy budowie algorytmów inteligencji obliczeniowej. Inteligentne systemy powinny mieć na celu: wybierać odpowiednie dane, wydobywać informacje z danych, wnioskować na podstawie uzyskanych informacji. Ustalanie tych celów może być procesem hierarchicznym, z wieloma celami podrzędnymi, tworzącymi plan działania

lub rozwiązanie problemu. Co prawda ludzie są bardzo elastyczni w poszukiwaniu alternatywnych rozwiązań, ale obecne metody inteligencji obliczeniowej koncentrują się na poszukiwaniu jednego najlepszego rozwiązania. Mózgi szukają alternatywnych rozwiązań wykorzystując wiele wyspecjalizowanych modułów, z których niektóre są stosowane tylko w sytuacjach bardzo nietypowych. Pojedyncza kolumna kory nowej zawiera wiele typów mikroukładów, które reagują w różny sposób na sygnały wejściowe [58]. Inne kolumny kory mogą łączyć te odpowiedzi w sposób hierarchiczny tworząc złożone ukryte cechy. Ostateczne decyzje są podejmowane wówczas, gdy odpowiednia liczba cech (utworzonych przy pomocy filtrów informacji) dostarczy odpowiedniej informacji [82].

Klucz do zrozumienia ogólnej inteligencji może zatem spoczywać w określeniu odpowiednich filtrów informacji, które uczynią uczenie możliwym. Takie filtry zostały opracowane przez procesy ewolucyjne. Procesy integracji [68] łączą te informacje w umysłowe reprezentacje wyższego poziomu. Filtry złożone z neuronalnych mikroobwodów odkrywają fonemy, sylaby, wyrazy w strumieniu dźwiękowym, czy też linie i krawędzie w strumieniu wizualnym, a “proces zbrylania” łączy sekwencje niższego poziomu we wzory wyższego poziomu, odkrywając skojarzenia, motywy i obiekty.

Inteligencja obliczeniowa stara się stworzyć uniwersalne systemy uczenia, ale organizmy biologiczne często wykazują wrodzone wzorce zachowań, które są przydatne jedynie w rzadkich, bardzo szczególnych sytuacjach. Modele, które nie działają dobrze na całych danych, ale działają poprawnie w niektórych określonych przypadkach mogą także być niezwykle użyteczne. Dlatego też zamiast bezpośredniej próby rozwiązania wszystkich problemów jednym algorytmem, dobrą strategią jest przekształcenie ich w wiele mniejszych zadań, które następnie zostaną zintegrowane przez odpowiedni moduł decyzyjny.

Uczenie oparte na tych pomysłach wymaga kilku elementów takich jak:

- Specyficzne filtry do wydobycia istotnych informacji z oryginalnych danych, tworzące przydatne cechy wsparcia;
- Różne zestawienia transformacji, które tworzą cechy wyższego rzędu, analizując przypadki w rozszerzonej przestrzeni cech;
- Modele procesów decyzyjnych w oparciu o cechy wyższego rzędu.

Poniżej dokonana zostanie analiza różnych transformacji, które wyodrębniając informacje tworzą nowe cechy. Kolejne podrozdziały prezentują korzyści jakie niesie rozszerzona przestrzeń cech dla uczenia opartego o transformacje, jak zdefiniować na nowo cel uczenia i jak przenieść wiedzę pomiędzy różnymi zadaniami uczenia.

6.2 Wydobywanie użytecznych cech

Rozpoznawanie obiektów przez mózg, jest wykonywane prawdopodobnie w oparciu o ocenę podobieństwa do zapamiętanych prototypów obiektów przy pomocy kilku charakterystycznych cech [90], ale dla różnych klas, cechy te mogą być zupełnie inne. Korzystanie z różnych cech w różnych regionach przestrzeni cech może znacznie uprościć problemy kategoryzacji.

Przestrzeń cech początkowych dla zbioru obiektów O określona jest przez miary podobieństwa do innych obiektów, tworząc wektor oryginalnych (surowych) danych

wejściowych ${}^0\mathbf{X}(O) = \mathbf{X}(O)$. Nauka z takich danych odbywa się przez szereg transformacji, które generują nowe cechy wyższego rzędu.

Wstępne transformacje $\mathcal{T}_1$ surowych danych powinny polepszać informacje niezbędne do realizacji celów uczenia. Na tym etapie łączenie małych podgrup cech za pomocą uczenia Hebbowskiego, opartego na korelacji, jest często bardzo przydatne. Nowy zestaw danych ${}^1\mathbf{X} = \mathcal{T}_1({}^0\mathbf{X})$ kształtuje obraz oryginalnych danych w przestrzeni nowych cech. W zależności od danych i celów uczenia, obszar ten może mieć wymiarowość, która jest mniejsza lub większa niż dane oryginalne.

Druuga transformacja ${}^2\mathbf{X} = \mathcal{T}_2({}^1\mathbf{X})$ wyodrębnia zwykle wielowymiarowe informacje ze wstępnie przetworzonych cech ${}^1\mathbf{X}$. Wymaga to oceny, która z możliwych transformacji na poziomie $\mathcal{T}_1$ może wydobyć informacje, które będą przydatne dla konkretnych transformacji $\mathcal{T}_2$. Wiele z tych aspektów może być brane pod uwagę podczas definiowania takich transformacji. Należy także pamiętać, że niektóre typy cech, nie są odpowiednie dla niektórych modeli uczenia się i procedur optymalizacji. Na przykład cechy binarne mogą nie działać dobrze z gradientowymi technikami optymalizacji, a normalizacja może przeszkadzać, jeśli wymagane są rozwiązania oparte na regułach.

Kolejne przekształcenia mogą używać $\mathcal{T}_2$ oraz $\mathcal{T}_1$, a także cech surowych. Proces jest powtarzany aż do ostatecznego przekształcenia, którego celem jest albo podział danych, albo mapowanie do konkretnej struktury, która może być łatwo rozpoznana przez dostępne algorytmy decyzyjne. Cechy wyższego rzędu stworzone po serii k przekształceń ${}^k\mathbf{X}_i$ powinny być traktowane w taki sam sposób, jak cechy surowe. Wszystkie cechy wpływają na kształt obszarów decyzji, a taka perspektywa pomaga zrozumieć ich zalety i ograniczenia. Proces uczenia powinien również uwzględniać koszty obliczeniowe różnych transformacji.

6.2.1 Wydobywanie informacji z pojedynczych cech

Wstępne transformacje danych mogą krytycznie wpływać na zbieżności algorytmów uczenia i budowę końcowych modeli danych. Surowe cechy wejściowe mogą zawierać użyteczne informacje, ale nie wszystkie algorytmy potrafią z niej skorzystać.

Metody obliczania odległości lub iloczynów skalarnych korzystają z normalizacji i standaryzacji wartości cech. Korzystanie z logarytmicznej, sigmoidalnej, wykładniczej, wielomianowej i innych prostych funkcji, tak by gęstość punktów w jednym wymiarze była bardziej jednolita, może czasem okazać się pomocne. Standaryzacja jest pomocna przy korzystaniu z cech do obliczania odległości (metody najbliższego sąsiada, większość metod kernelowych, sieci RBF), ale również pomaga przy inicjalizacji wag w kombinacjach liniowych (dyskryminacja liniowa, MLP), natomiast nie jest potrzebna do reguł logicznych, ani do drzew decyzyjnych.

6.2.2 Cechy binarne

Cechy binarne B_i są najprostsze i wskazują na obecność lub brak pewnych obserwacji. Mogą one również być tworzone przez podzielenie nominalnej cechy na podzbiory, lub tworzenie podprzedziałów cech rzeczywistych. Korzystając z metod filtrów [20], metod dyskretyzacji lub takich algorytmów jak 1R [45], przedziały wartości cech rzeczywistych, które są skorelowane z wyjściem mogą być wybrane i przedstawione jako cechy binarne. Z geometrycznego punktu widzenia cecha binarna to etykieta wyróżniająca dwa podzbiory, umieszczająca wszystkie wektory w punkcie 0 lub 1 na linii współrzędnych.

Wartości cech są zwykle definiowane globalnie, dla wszystkich dostępnych danych. W określonym kontekście niektóre cechy są użyteczne tylko lokalnie. Z geometrycznego punktu widzenia są rzutami wektorów, które należą do podprzestrzeni, w których spełnione są szczególne warunki i powinny pozostać nieokreślone dla wszystkich innych wektorów. Zwiększenie przestrzeni cech poprzez dodanie zdefiniowanych warunkowo cech może znacząco wpłynąć na poprawę wyników.

Zdefiniowane warunkowo cechy binarne można uzyskać poprzez nałożenie różnych ograniczeń na podprzestrzenie wektorów stosowane do rzutów. Zamiast używać bezpośrednio surowych cech, można skorzystać z warunków $B_i = T \wedge LT_i(\mathbf{X})$ oraz $B_i = F \wedge LF_i(\mathbf{X})$, gdzie $LT(\mathbf{X}), LF(\mathbf{X})$ to funkcje logiczne określające ograniczenia przy użyciu wektora cech $\mathbf{X}$.

6.2.3 Cechy z wartościami rzeczywistymi

Z geometrycznego punktu widzenia wartości rzeczywiste cech wejściowych uzyskane z różnych badań i pomiarów na zestawie obiektów są projekcją na prostą o pewnych właściwościach.

Pojedyncze cechy mogą wykazać ciekawe struktury wśród rozkładów $p(C|X)$, na przykład rozkłady periodyczne, lub k czystych klastrów. Rzuty na linię, które pokazują rozkłady k -separowalnych danych są bardzo przydatne do nauki skomplikowanych funkcji logicznych. Przykładowo dla problemu parzystości n -bitowej można wyróżnić $n + 1$ oddzielnych klastrów w rzutach na najdłuższą przekątną, tworząc w ten sposób nowe przydatne cechy (patrz rys. 4.8).

Dla cech o wartościach rzeczywistych, najprostsze ograniczenia można wykonać przez iloczyn przedziałów $\prod_i [r_i^-, r_i^+]$, lub iloczyny bicentralnych funkcji tworzących hiperkostki [25], w których powinny leżeć rzutowane wektory. Chodzi w tym wszystkim o to, aby przekształcone funkcje nadawały etykiety różnym rejonom przestrzeni cech, upraszczając analizę danych w tych rejonach.

6.2.4 Projekcje liniowe

Grupa kilku skorelowanych cech może być zastąpiona przez ich pojedynczą kombinację, np. przy wykorzystaniu analizy składników głównych. Kombinacje liniowe silnie skorelowanych cech pozwalają nie tylko na zmniejszenie liczby wymiarów, ale również na tworzenie cech w różnych skalach, łączenie kilku cech, czy stworzenie globalnej kombinacji PCA wszystkich cech. Takie podejście może przyczynić się do odkrycia hierarchicznych zestawów cech, które są przydatne w wielu problemach. Innym sposobem uzyskania cech jest wykonanie klasteryzacji w przestrzeni danych wejściowych i wykonanie lokalnego PCA w ramach klastrów, aby znaleźć cechy, które są najbardziej przydatne lokalnie w różnych obszarach przestrzeni.

Można wykorzystać wiele transformacji nadzorowanych do określenia współczynników kombinacji cech wejściowych:

- Dowolna miara zależności między klasą i rozkładem wartości cech, taka jak współczynnik korelacji Pearsona, χ^2 , kryterium separowalności [39];
- Miary oparte na informacji, takie jak informacja wzajemna pomiędzy etykietami a nowymi cechami [86];

- Liniowa analiza dyskryminacyjna (LDA), z każdą nową cechą bazującą na kierunku ortogonalnym do tego otrzymanego z LDA czy liniowego SVM [31, 91];
- Analiza dyskryminacyjna Fishera (FDA) [91];
- Liniowa analiza czynnikowa [37];
- Zlokalizowane rzuty czystych klastrów za pomocą różnych indeksów poszukiwania projekcji (*projection pursuit*) [50, 35].

6.2.5 Cechy kernelowe

Najbardziej popularny rodzaj algorytmu SVM ze zlokalizowanymi (zwykle gaussowymi) kernelami [80] cierpi z powodu przekleństwa wymiarowości [5]. Wynika to z faktu, że takie algorytmy opierają się na założeniu jednolitej rozdzielczości i lokalnym podobieństwie próbek danych. Aby uzyskać dokładne rozwiązanie, często wymagana jest duża liczba przykładów uczących, wykorzystanych jako wektory wsparcia. Prowadzi to do wysokich kosztów obliczeń i skomplikowanych modeli, które nie dokonują dobrej generalizacji. Poświęcono wiele wysiłku na doskonalenie skalowania [88, 10], zmniejszenie liczby wektorów wsparcia [76] i uczenia wielu kerneli [84]. Wszystkie te osiągnięcia są imponujące, ale wciąż istnieje przestrzeń dla sposobów prostszych, bardziej bezpośrednich i zrozumiałych.

Ogólnie im większy wymiar przekształconej przestrzeni, tym większa szansa, że dane mogą być rozdzielone przez hiperpłaszczyzny [44]. Jeden z popularnych sposobów tworzenia wysoko-wymiarowej reprezentacji bez zwiększania kosztów obliczeniowych polega na stosowaniu kernelowego triku [80]. Szuka się następnie rozwiązania opartego na dyskryminacji liniowej (LDA) w nowej przestrzeni cech ${}^2X = \mathbf{K}(\mathbf{X}) = K({}^1\mathbf{X}, \mathbf{X})$. Metody kernelowe działają, ponieważ pośrednio dostarczają one nowych, przydatnych cech $Z_i(\mathbf{X}) = K(\mathbf{X}, \mathbf{X}_i)$ skonstruowanych przy użyciu wektorów wsparcia $\mathbf{X}_i$. Liniowe rozwiązania SVM w przestrzeni cech kernelowych $\mathbf{Z}$ są równoważne rozwiązaniom SVM, co sprawdzono empirycznie [65] (por. tabela 5.8).

Techniki selekcji cech mogą być stosowane do pozostawienia tylko składników odpowiadających “wektorom wsparcia”, które zapewniają podstawowe wsparcie dla klasyfikacji. Na przykład (w zależności od rodzaju problemu) tylko tych, które są w pobliżu granic decyzyjnych lub tych w pobliżu centrów klastrów. Możliwe jest także użycie dowolnej metody inteligencji obliczeniowej w kernelowej przestrzeni cech $K(\mathbf{X})$. Jest to idea, którą wykorzystuje algorytm maszyn cech wspierających [65]. Jeśli wymiarów jest dużo, istnieje duża szansa przeuczenia się, dlatego powinny być używane tylko najprostsze i najbardziej niezawodne modele. Dobrym rozwiązaniem wydaje się być tutaj użycie LDA z maksymalizacją marginesu, a więc popularnego SVMa z kernelem liniowym.

Jawne generowanie cech w oparciu o różne miary podobieństwa [77] usuwa jedno z ograniczeń SVM, pozwalając na optymalizację rozdzielczości w różnych obszarach przestrzeni cech, zapewniając silne nieliniowości tam, gdzie są potrzebne (małe dyspersje w funkcji Gaussa) i wykorzystując wygładzone funkcje, jeżeli jest to wystarczające. Technika ta może być nazwana adaptacyjną regularyzacją, w przeciwieństwie do prostej regularyzacji opartej o minimalizację normy wektora wag $\|\mathbf{W}\|$ stosowanej w SVM lub sieciach neuronowych. Mimo, że prosta regularyzacja wymusza gładkie granice decyzyjne, zmniejszając złożoność modelu nie jest ona w stanie znaleźć najprostszych

rozwiązań i może łatwo przeoczyć fakt, że np. jedna cecha binarna zawiera wszystkie informacje niezbędne do rozwiązania problemu.

Generowanie cech kernelowych powinno zatem postępować od najbardziej ogólnych, umieszczonych z dala od granicy decyzji (takie wektory można łatwo zidentyfikować, patrząc na rozkład $z = \mathbf{W} \cdot \mathbf{X}$ dla $\mathbf{W} = (\mathbf{m}_1 - \mathbf{m}_2)/\|\mathbf{m}_1 - \mathbf{m}_2\|$).

Cechy kernelowe oparte na funkcjach radialnych, to projekcje na jeden wymiar radialny i w tym sensie podobne są one do liniowej projekcji. Jednak liniowe projekcje mają charakter globalny i są niezależne od pozycji, podczas gdy projekcje radialne wykorzystują wektor odniesienia $K(\mathbf{X}, \mathbf{R}) = \|\mathbf{X} - \mathbf{R}\|$, który pozwala na skupienie się na obszarze blisko $\mathbf{R}$. Dodatkowo potrzebne są czynniki skalowania w celu uwzględnienia znacząco różnych cech $K(\mathbf{X}, \mathbf{R}; \mathbf{W}) = \|\mathbf{W} \cdot (\mathbf{X} - \mathbf{R})\|$. Jeśli stosuje się kernele gaussowskie, prowadzi to do cech typu $G(\mathbf{W}(\mathbf{X} - \mathbf{R}))$.

6.2.6 Inne nieliniowe mapowania

Kombinacje liniowe pochodzące z ciekawych kierunków projekcji mogą zapewnić pewną liczbę interesujących cech, ale w niektórych zastosowaniach niezbędne jest nieliniowe przetwarzanie danych. Liczba możliwych transformacji w takim przypadku jest bardzo duża. Np. sieci neuronowe używające funkcji wymiernych [44] (wykorzystywane przy przetwarzaniu sygnałów [56]) używają wielomianowych kombinacji cech.

Przekształcenia liniowe w przestrzeni kernelowej są równoważne nieliniowym transformacjom w przestrzeni cech oryginalnych. Kilka nieliniowych transformacji wymieniono poniżej:

- Kernelowe wersje przekształceń liniowych [80];
- Transformacje oparte na ważonych odległościach [77];
- Perceptrony z funkcjami sigmoidalnymi, takie jak w warstwach sieci MLP;
- Transformacje heterogeniczne przy użyciu kilku rodzajów kerneli aby zapewnić wielorozdzielczość [25];
- Skalowanie wielowymiarowe (MDS) w celu zmniejszenia wymiarów przy zachowaniu odległości [13].

6.2.7 Podsumowanie rodzajów nowych cech

Cechy mogą być ważne i łączone za pomocą funkcji odległości, kerneli, ukrytych warstw i wielu innych, ale geometryczna perspektywa pokazuje, jakiego rodzaju informacje mogą być z nich pozyskane. Próbę skategoryzowania różnych typów cech z tej perspektywy, w tym warunkowo określonych cech, pokazano poniżej. X oznacza tu dowolny typ cechy skalarnej, B to cecha binarna, N nominalna, R cecha z ciągłymi wartościami rzeczywistymi, K to ogólna cecha kernelowa, M to motywy i sekwencje a S to sygnały.

- B1) Binarne, równoważne nieograniczonym projekcjom na dwa punkty;
- B2) Binarne, ograniczone przez inne cechy binarne, kompleksy $B_1 \wedge B_2 \dots \wedge B_k$;

- B3) Binarne, projekcje podprzestrzeni ograniczone przez odległość $B = 0 \wedge R_1 \in [r_1^-, r_1^+]\dots \wedge R_k \in [r_k^-, r_k^+]$;
- N1–N3) Nominalne cechy są podobne do binarnych, z podzbiorami zamiast przedziałów;
- R1) Rzeczywiste, równoważne nieograniczonym ortogonalnym rzutom na prostą, z progami i skalowaniem;
- R2) Rzeczywiste, ortogonalne rzuty na prostą, ograniczone przedziałami, obejmujące podprzestrzenie prostopadłe do rzutu;
- R3) Rzeczywiste, ortogonalne rzuty z cylindrycznymi ograniczeniami odległości od prostej;
- R4) Rzeczywiste, dowolna zoptymalizowana projekcja na prostą (PCA, ICA, LDA);
- R5) Rzeczywiste, dowolna projekcja na linię z okresowymi lub pół-okresowymi przedziałami lub prawdopodobieństwa *a posteriori* dla każdej klasy, liczone wzdłuż prostej $p(C|X)$;
- K1) Cechy kernelowe $K(\mathbf{X}, \mathbf{R}_i)$ z wektorami referencyjnymi $\mathbf{R}_i$;
- K2) Cechy kernelowe z przedziałami;
- M1) Motywy, bazujące na korelacji pomiędzy elementami i na sekwencjach symboli;
- S1) Dekompozycja sygnału i projekcje na funkcje bazowe;
- T1) Inne transformacje nieliniowe ograniczające podprzestrzenie w sposób bardziej złożony, funkcje wymierne, uniwersalne funkcje transferu [25].

Możliwe są również kombinacje różnych rodzajów cech, na przykład cylindryczne ograniczenia z przedziałami. Zestawienie podane powyżej nie jest bardzo precyzyjne i dalekie od kompletnego, ale powinno dać pojęcie jaki rodzaj granic decyzji może być generowany przez różne typy cech. Cechy wyższego rzędu mogą być budowane przez maszyny uczące przy użyciu cech, które zostały zbudowane przez wcześniejsze transformacje. Wskaźniki istotności tych cech lub metody selekcji cech, powinny pomóc określić ich znaczenie, chociaż niektóre cechy mogą być wymagane tylko do lokalnego przedstawienia informacji, więc ich ogólne znaczenie może być niskie [20].

6.3 Lokalnie optymalizowane kernele

Zaprezentowana w tym rozdziale metoda lokalnie optymalizowanych kerneli (*Locally Optimized Kernels*, *LOK*) opiera się na generacji nowych cech przy użyciu ograniczonych kerneli gaussowskich.

Aby utworzyć nową przestrzeń cech, algorytm LOK dla każdego wektora treningowego tworzy tymczasową kernelową cechę gaussowską: $G_i(\mathbf{X}) = \exp(-||\mathbf{X}_i - \mathbf{X}||^2/2\sigma^2)$. Dla każdej takiej cechy G_i przeanalizowany zostaje rozkład $p(G_i|C)$ w celu znalezienia czystych klastrów w pewnym przedziale $I_{iab} = [g_{ia}, g_{ib}]$. Prowadzi to do

utworzenia potencjalnych cech binarnych $B_{iab}(\mathbf{X})$. Dobry kandydat na cechę powinien zawierać co najmniej η wektorów treningowych. Jeśli warunek jest spełniony, cecha taka jest dodawana do puli zestawu nowych, użytecznych cech. Poniżej parametr η został zoptymalizowany przy pomocy wewnętrznej krosvalidacji. Aby wykorzystać algorytm LOK w charakterze klasyfikatora i w ten sposób wykazać istotność i jakość generowanych nowych atrybutów, do znalezienia rozwiązania w nowej przestrzeni cech używany jest mechanizm WTA lub liniowa dyskryminacja.

Schemat 5 Lokalnie optymalizowane kernele

Require: Ustaw wartości wewnętrznych parametrów: η dla minimalnego pokrycia i σ dla dyspersji.

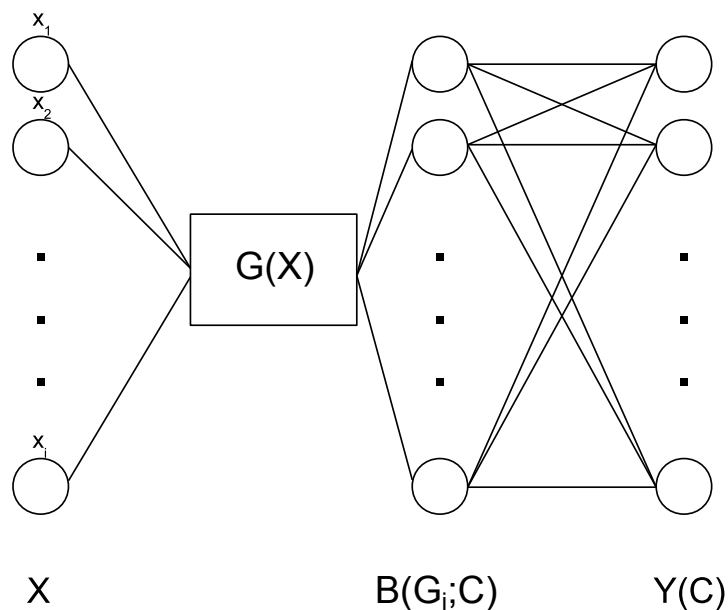
- 1: Zestandaryzuj dane, m wektorów, d cech.
 - 2: Utwórz potencjalne, kernelowe cechy gaussowskie $G_i(\mathbf{X}) = \exp(-\|\mathbf{X}_i - \mathbf{X}\|^2/2\sigma^2)$.
 - 3: Posortuj wartości $G_i(\mathbf{X})$ malejąco, wraz z przypisanymi etykietami klas.
 - 4: Przeanalizuj rozkład $p(G_i|C)$ aby znaleźć wszystkie interwały z czystymi klastrami, wyznaczającymi cechy binarne $B_{iab}(\mathbf{X}; C)$.
 - 5: **if** liczba wektorów pokryta przez cechę $B_{iab}(\mathbf{X}; C) > \eta$ **then**
 - 6: Zaakceptuj cechę binarną tworząc opisany przez klasę, węzeł w ukrytej warstwie sieci.
 - 7: **end if**
 - 8: Sklasyfikuj dane testowe zmapowane do rozszerzonej przestrzeni:
 - 9: Zsumuj aktywność ukrytych węzłów dla każdej z klas, aby wyznaczyć wyjścia sieci (WTA), lub
 - 10: Zbuduj liniowy model na rozszerzonej przestrzeni cech (LDA).
-

W algorytmie LOK są dwa parametry, które należy ustawić: η wyznaczający minimalną wielkość klastra (ilość wektorów na klaster), oraz σ kontrolujący dyspersję zlokalizowanych cech gaussowskich. Czyste klastry są znajduwane zarówno w lokalnym sąsiedztwie wektorów wsparcia w przedziale $[a, b]$, ($a = 0$), lub jeśli wektor wsparcia jest otoczony przez wektory z innej klasy, mogą się one znaleźć nieco dalej (wówczas a będzie przyjmować wartości $0 < a < b$). Czyste klastry i cechy binarne mogą być całkiem użyteczne przy identyfikacji regionów z wektorami, które mogą zostać sklasyfikowane z dużym współczynnikiem zaufania. Dla bardzo dużych danych takie wektory mogą zostać usunięte, pozostawiając jedynie obszary znajdujące się blisko granic decyzji. Tak więc LOK może się w tym przypadku znakomicie sprawdzić w roli eliminatora (patrz podrozdział 5.1.1).

Dla każdego potencjalnego wektora wsparcia G_i , dla którego znaleziono $p(G_i|C) = p(G_i|\neg C)$ przy dyspersji wynoszącej $\sigma_i = b/2$, tworzy się nową, kernelową cechę gaussowską $G_i(\mathbf{X}; b) = \exp(-\|\mathbf{X}_i - \mathbf{X}\|^2/b)$. W pewnych przypadkach większa liczba cech wsparcia może zostać utworzona przy pomocy dużego σ_i i analizy rozkładów $p(G_i|C)$ dla wartości większych niż b , używając interwałów $I_{iab} = [G_{ia}, G_{ib}]$, ($a > 0$), gdzie jedna z klas dominuje. Te nowe cechy otrzymuje się jako różnicę dwóch funkcji gaussowskich $G_i(\mathbf{X}; b) - G_i(\mathbf{X}; a)$.

W celu znalezienia rozwiązania w nowej przestrzeni cech, LOKLDA używa liniowej dyskryminacji z maksymalizacją marginesu (optymalny margines jest wybierany przy użyciu wewnętrznej krosvalidacji).

Cechy utworzone przez LOK mogą zostać użyte jako węzły sieci, które reprezentują kernelowe transformacje. Dodatkowe warstwy sieci są następnie używane do analizy danych w ten sposób zdefiniowanej przestrzeni cech (patrz rys. 6.1).



Rysunek 6.1: Algorytm LOK przedstawiony w postaci sztucznej sieci neuronowej.

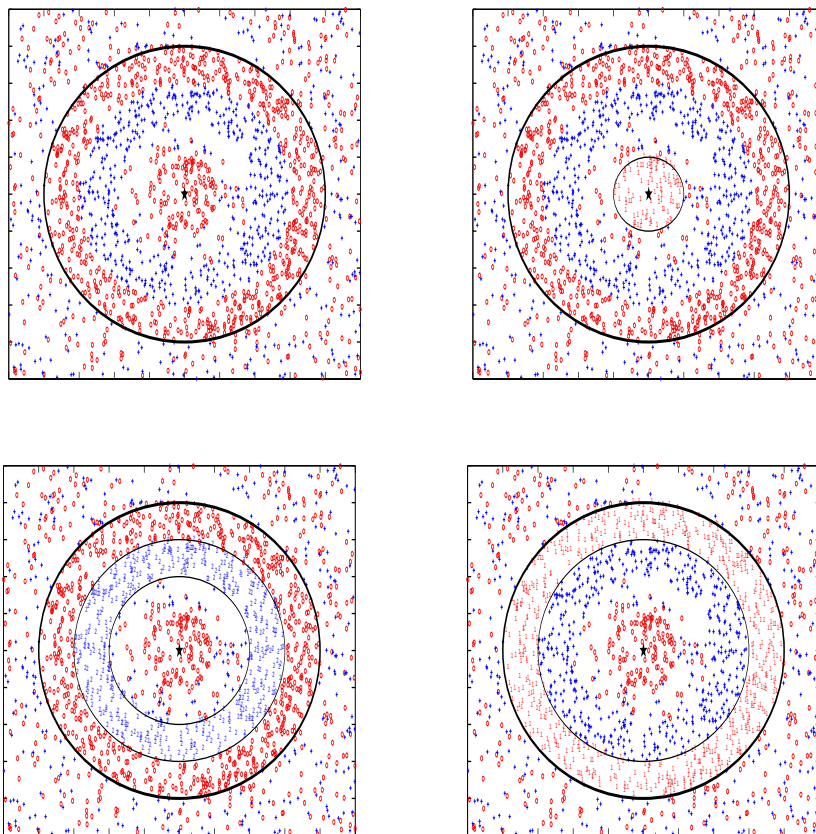
Aby pokazać efekt optymalizacji kerneli, przeprowadzono porównanie czterech metod: standardowego SVM z kernelem liniowym (SVML) i z kernelem gaussowskim (SVMG), LOK z mechanizmem WTA (LOKWTA), oraz LOK w połączeniu z liniową dyskryminacją (LOKLDA). Algorytmy te zostały przetestowane na 27 zbiorach danych pobranych z UCI Machine Learning Repository [3]. Przypadki z brakującymi wartościami cech (jeśli występowały) zostały usunięte.

Wyniki zgromadzone w tabeli 6.1 (najlepsze rezultaty zaznaczono pogrubioną czcionką) to uśrednione wartości poprawności klasyfikacji i odchyłeń standardowych, otrzymane z 10 razy powtórzonej 10-krotnej krosvalidacji. Parametry SVMów (C i σ) zostały zoptymalizowane przy użyciu wewnętrznej krosvalidacji.

Rezultaty przy użyciu LOK są w niemal każdym przypadku (poza jednym) statystycznie nie gorsze od tych uzyskanych przez SVM, a wręcz dla większości zbiorów lepsze. Jak można było przewidzieć, LOKWTA osiągnął najlepszy rezultat tylko w kilku (konkretnie to 4) przypadkach. SVML także osiągnął najlepszy wynik w 4 przypadkach. Dla 11 zbiorów danych LOKLDA pobił wszystkie inne metody, podczas gdy SVMG był najlepszy w 8 przypadkach, aczkolwiek tylko przy jednym zbiorze różnica okazała się statystycznie istotna.

Wyniki te pokazują, że w większości przypadków LOK działa lepiej niż oryginalny SVM. LOK wydaje się być także prostą metodą generacji cech, które mogą być użyte do poprawy oryginalnej SVMowej przestrzeni kernelowej.

Algorytm opisany w tym podrozdziale skupia się na generacji nowych, użytecznych, ograniczonych cech kernelowych. Pokazano że LOK jest całkiem użyteczną i prostą metodą generowania takich cech, które mogą zostać użyte do budowy lepszej kernelowej przestrzeni cech w algorytmie SVM. Podczas kiedy dużo wysiłku kładzie się na polepszenie algorytmów uczenia metody SVM, znacznie mniej uwagi skupia się na metody ekstrakcji informacji z danych, które powodują że liniowa dyskryminacja staje się prostsza i bardziej dokładna. W przypadku gaussowskich cech kernelowych, nie ma powodu dlaczego używać takiej samej dyspersji dla wszystkich wektorów wsparcia. Te wektory wsparcia, które są daleko od granicy decyzji po złej stronie, powinny dostarczyć cech kernelowych, które wykluczać będą lokalne sąsiedztwo, podczas kiedy takie same



Rysunek 6.2: Potencjalnie użyteczna gaussowska cecha kernelowa dla radialnych rozkładów danych oraz trzy nowe użyteczne cechy utworzone za pomocą algorytmu LOK.

wektory wsparcia, ale znajdujące się po właściwej stronie powinny używać raczej dużych dyspersji. Wektory wsparcia blisko granic decyzji powinny cechować się raczej małymi dyspersjami. Algorytm LOK tworzy takie lokalnie optymalizowane kernele gaussowskie i może być łatwo połączony z różnymi metodami selekcji cech w celu lepszego działania.

6.4 Uczenie oparte na transformacjach

Koniecznym krokiem dla uczenia opartego na transformacjach jest stworzenie taksonomii, kategoryzującej i opisującej podobieństwa i relacje między transformacjami oraz ułatwienie systematycznego poszukiwania tych przemian w przestrzeni wszystkich możliwych kompozycji. Oczwistym wydaje się być podział między transformacjami zoptymalizowanymi lokalnie, z dobrze zdefiniowanymi celami i transformacjami adaptacyjnymi, które są oparte na dalszych kryteriach, gdzie cele są określane globalnie (jak we wstecznej propagacji).

W pierwszym przypadku aktywność węzłów sieci odpowiadających za transformacje ma jasną interpretację, a rosnąca złożoność dodawania nowych węzłów powinna być uzasadniona przez odkrywanie nowych aspektów danych. Lokalne transformacje $\mathcal{T}_2$ mają współczynniki obliczane bezpośrednio z danych wejściowych lub danych po transformacji $\mathcal{T}_1$. Mogą być one bardzo przydatne dla inicjowania globalnych

Dane	SVML	SVMG	LOKWTa	LOKLDA
Arrhythmia	50.92±17.31	43.36±21.47	42.00±24.19	39.10±12.98
Autos	54.48±13.75	74.29±12.58	58.69±11.03	74.36±10.40
Balance-scale	84.47±3.17	89.83±2.09	90.71±2.38	96.46±2.62
Breast-cancer	73.27±6.10	75.67±5.35	76.58±6.37	75.09±1.99
Breast-w	96.60±2.07	96.77±1.84	96.93±1.62	97.21±2.13
Car	67.99±2.61	98.90±0.90	84.72±3.44	93.57±1.81
Cmc	19.14±2.14	34.09±3.67	48.54±2.52	51.06±4.30
Credit-a	86.36±2.86	86.21±2.90	82.67±4.01	84.70±4.91
Credit-g	73.95±4.69	74.72±4.03	73.10±2.38	72.70±3.86
Cylinder-bands	74.58±5.23	76.89±7.57	74.32±6.41	80.11±7.53
Dermatology	94.01±3.54	94.49±3.88	87.97±5.64	94.71±3.02
Diabetes	76.88±4.94	76.41±4.22	74.88±3.88	76.95±4.47
Ecoli	78.48±5.90	84.17±5.82	82.47±3.66	85.66±5.40
Glass	42.61±10.05	62.43±8.70	64.96±7.72	71.08±8.13
Haberman	72.54±1.96	72.91±5.93	76.46±4.34	73.53±0.72
Heart-c	82.62±6.36	80.67±7.96	81.07±7.56	81.04±5.17
Heart-statlog	83.48±7.17	83.40±6.56	81.48±8.73	83.33±7.46
Hepatitis	83.25±11.54	84.87±11.98	89.88±10.14	84.05±4.40
Ionosphere	87.72±4.63	94.61±3.68	85.18±6.28	95.16±2.72
Iris	72.20±7.59	94.86±5.75	94.67±6.89	93.33±5.46
Kr-vs-kp	96.03±0.86	99.35±0.42	83.73±2.58	98.25±0.45
Liver-disorders	68.46±7.36	70.30±7.90	57.40±5.72	69.72±6.57
Lymph	81.26±9.79	83.61±9.82	76.96±13.07	80.52±7.91
Sonar	73.71±9.62	86.42±7.65	86.57±7.01	86.52±8.39
Vote	96.12±3.85	96.89±3.11	92.57±7.52	93.95±4.18
Vowel	23.73±3.13	98.05±1.90	92.49±3.37	97.58±1.52
Zoo	91.61±6.67	93.27±7.53	88.47±5.35	94.07±6.97

Tabela 6.1: Rezultaty klasyfikacji 26 zbiorów danych przy użyciu algorytmu LOK.

transformacji adaptacyjnych lub mogą być użyteczne dla znalezienia lepszych rozwiązań bardziej złożonych transformacji. Na przykład, skalowanie wielowymiarowe wymaga bardzo trudnej minimalizacji i przeważnie szybciej się zbiega, jeśli na początku zastosuje się transformację PCA.

Po wstępnych transformacjach całe dane są konwertowane na wewnętrzną reprezentację ${}^k\mathbf{X}$, tworząc nowy obraz danych, rozłożonych w prostszy sposób niż obraz oryginalny. Ostateczna transformacja powinna być w stanie wyodrębnić żądane informacje z tego obrazu. Jeśli ostateczna transformacja jest liniowa $\mathbf{Y} = {}^{k+1}\mathbf{X} = \mathcal{T}_{k+1}({}^k\mathbf{X}; {}^k\mathbf{W})$, parametry ${}^k\mathbf{W}$ są albo określone w procedurze iteracyjnej jednocześnie ze wszystkimi innymi parametrami $\mathbf{W}$ z poprzednich transformacji (tak jak w algorytmach wstecznej propagacji [44]), albo kolejno wyznacza się je przez obliczenie transformacji pseudo-odwrotnej, jak to często praktykuje się w dwu-fazowym uczeniu RBF [81].

Trzy podstawowe strategie tworzenia zestawu transformacji to:

- Użycie metody konstruktywistycznej przez dodawanie cech na podstawie prostych transformacji, kontynuowane dotąd, gdy wzrost dokładności nie uzasadni wzrostu złożoności [48, 15];
- Start od złożonych transformacji i optymalizacja parametrów, na przykład za pomocą elastycznych, neuronowych funkcji transferu [19, 74], optymalizując każdą transformację przed dodaniem kolejnej;
- Technika przycinania i regularyzacji dla dużych sieci z węzłami opartymi o proste przekształcenia i optymalizację globalną [44].

Po dodaniu każdej nowej cechy, obraz danych w rozszerzonej przestrzeni cech zmienia się i nowe transformacje powstają w tej przestrzeni, a nie w oryginalnej. Na przykład, dodając transformacje z warunkami opartymi o odległości można dodać nowe cechy kernelowe i zacząć budowanie ostatecznej transformacji, przypisując istotne wagi tylko do cech wsparcia opartych na kernelach.

Z geometrycznego punktu widzenia transformacje kernelowe są w stanie wygładzić lub spłaszczyć granice decyzji za pomocą wektorów wsparcia $\mathbf{R}$, które leżą blisko złożonych granic decyzji. Pojedyncza hiperpłaszczyzna po takiej transformacji jest często wystarczająca, aby osiągnąć dobrą separację danych.

Metody kernelowe rozszerzają wymiarowość przestrzeni cech, jeśli liczba próbek jest większa niż liczba cech wejściowych (por. neurobiologiczne uzasadnienie takich rzutów w [58]). Rozszerzenie wymiarowości danych zwiększa szansę, aby były one separowalne, a to jest często celem tej transformacji ${}^2\mathbf{X} = \mathcal{T}_2({}^1\mathbf{X}; {}^1\mathbf{W})$. Losowe liniowe projekcje wektorów wejściowych w przestrzeń wysokowymiarową ${}^2\mathbf{X} = \mathbf{L}({}^1\mathbf{X})$ to najprostszy sposób na zwiększenie wymiarów, z losową macierzą $\mathbf{L}$, która ma więcej wierszy niż kolumn. Ostateczna transformacja jest przekształceniem liniowym $\mathbf{Y} = \mathcal{T}_3({}^2\mathbf{X}; {}^2\mathbf{W}) = \mathbf{W}_2 \cdot {}^2\mathbf{X}$, choć może ona nie być optymalnym (najprostszym lub najdokładniejszym) rozwiązaniem.

6.4.1 Przedefiniowanie celu uczenia

Zazwyczaj systemy eksploracji danych zwracają tylko jeden najlepszy model, ale jeśli optymalizowanych jest kilka kryteriów, lepszym wydaje się być znalezienie zestawu Pareto optymalnych modeli. Na przykład dokładność powinna być maksymalizowana, ale wariancja powinna być zminimalizowana lub czułość powinna zostać zmaksymalizowana, a ilość fałszywych alarmów powinna być poniżej pewnego progu.

Każdy model danych opiera się na pewnych szczególnych założeniach dotyczących rozkładu danych w przestrzeni wejściowej i może być stosowany z powodzeniem tylko do niektórych rodzajów problemów. Na przykład SVM i wiele innych statystycznych metod uczenia [80] polega na założeniu jednolitego rozkładu, czy lokalnym podobieństwie między próbkami danych. Takie metody mogą nie poradzić sobie z wysoko wymiarowymi funkcjami, które nie są wystarczająco gładkie [5]. W takim przypadku dokładne rozwiązanie może wymagać bardzo dużej liczby próbek uczących, które będą wykorzystane jako wektory odniesienia, co prowadzi do wysokich kosztów obliczeń oraz tworzenia złożonych modeli, które mogą niewystarczająco generalizować. Równoległe

strumienie hierarchicznych transformacji, które odkrywają interesujące aspekty danych, koncentrują się na lokalnych ulepszeniach, a nie na jakimś globalnym celu, takim jak rozseparowanie danych. Obraz oryginalnych danych w rozszerzonej przestrzeni może zawierać pewne charakterystyczne wzorce, które procesy decyzyjne powinny wychwycić.

Jednym ze sposobów aby odkryć, jaki rodzaj struktury pojawia się po transformacji danych jest użycie wizualizacji danych w oryginalnej i rozszerzonej przestrzeni cech [63, 64]. Rzuty PCA mogą wykazywać ciekawe struktury w danych. Skalowanie wielowymiarowe (MDS) [13] jest nieliniowym odwzorowaniem, które stara się wiernie wyświetlać odległości między wektorami. Przydatne są również projekcje oparte na kierunkach uzyskanych z liniowego SVM. Analiza dyskryminacyjna Fishera (FDA) jest kolejną metodą, która może być wykorzystana do wizualizacji [91].

Wizualizacja transformacji w przypadku trudnych problemów logicznych odsłania naturę trudności danego problemu i pomaga wyznaczyć łatwiejsze cele uczenia. Rozważając problem parzystości: każdy wektor oznaczony jako parzysty jest otoczony przez wektory oznaczone jako nieparzyste i odwrotnie [26]. Zlokalizowane transformacje nie są w stanie uogólniać takich informacji, ale liniowe projekcje mogą dostarczyć ciekawe spojrzenie na te dane. Dla n -bitowej parzystości rzut $y = \mathbf{W} \cdot \mathbf{X}$, $\mathbf{W} = [1, 1 \dots 1]$, zlicza liczbę bitów, tworząc naprzemienne klastry z wektorami, które należą do klas parzystych i nieparzystych. Funkcja okresowa (taka jak kosinus) rozwiązuje problem parzystości za pomocą jednego parametru, ale nie podoła innym problemom logicznym. W przypadku wielu funkcji Boolowskich znalezienie przekształceń, które prowadzą do k -oddzielnych skupień, z wektorami jednej klasy w odstępach $[y_i, y_{i+1}]$ wzdłuż linii rzutu jest celem o wiele prostszym niż osiągnięcie separowalności. Cała przestrzeń cech jest wówczas podzielona na "równoległe plasterki", prostopadłe do kierunku $\mathbf{W}$.

Tak więc w celu utworzenia użytecznych cech powinno się znaleźć takie transformacje, które uchwycą częste korelacje wejść (uczenie Hebbowskie, PCA, odkrywanie motywów), lub odkryją klastry stosunkowo czystych danych używając do tego liniowych lub radialnych projekcji. Wizualizacja takiego obrazu danych powinna ujawnić, jakiego rodzaju metody są najbardziej odpowiednie do dalszej analizy.

6.4.2 Transfer wiedzy

Warto przyjrzeć się, kiedy różne algorytmy mogą sobie dobrze poradzić z danym problemem, a kiedy nie. Dane o prostej strukturze logicznej wymagają ostrych granic decyzji, oferowanych przez drzewa decyzyjne i systemy oparte na regułach decyzyjnych [27, 21], ale są dość trudne do analizy za pomocą algorytmów statystycznych lub neuronowych. SVM pomija proste rozwiązania w miejscu, gdzie najlepszą odpowiedź otrzymuje się np. przy pomocy jednej cechy binarnej. Często dane mają rozkład gaussowski i liniowa dyskryminacja (liniowy SVM, proste sieci MLP) daje najlepsze rozwiązanie. kNN i SVM w formie kernelowej działają dobrze, gdy granice decyzji mają złożoną topologię, ale nie poradzą sobie przy ostrych granicach decyzji lub gdy struktura danych ma złożoną budowę logiczną [40].

Jedną z często używanych przez ludzi strategii jest uczenie się bezpośrednio od innych. Transfer wiedzy skupia się na uczeniu wielu zadań związanych ze sobą na raz. W analizie obrazu, tekstu lub robotyce opracowanych zostało wiele metod transferu wiedzy. Obecne podejścia do transferu wiedzy skupiają się na wykorzystaniu dodatkowych danych, w celu stworzenia lepszego modelu dla określonych danych treningowych. Wykorzystywana jest ta sama przestrzeń cech i ten sam algorytm uczenia. Ten rodzaj

transferu wiedzy nie nadaje się do prezentowanego w tej rozprawie problemu. W naszym algorytmie Uniwersalnych Maszyn Uczących [30] transfer wiedzy pomiędzy różnymi algorytmami dokonywany jest przez udostępnianie nowych cech wyższego rzędu, które odniosły sukces w odkryciu pewnej użytecznej wiedzy w jednym z tych algorytmów. Drzewa decyzyjne i algorytmy oparte o reguły dostarczają cech binarnych (typu B1-B3). Cechy z wartościami rzeczywistymi (typu R1-R4) odkrywane są przez algorytmy pościgu projekcji (*projection pursuit*), liniowego SVM i prostych projekcji na linie łączące centra lokalnych klastrów. Naiwny klasyfikator Bayesa umożliwia nam wyznaczenie w tym kierunku prawdopodobieństw *a posteriori* $p(C|\mathbf{X})$. Metody kernelowe i kNN znajdują dobre cechy kernelowe bazując na podobieństwie. Najlepsze cechy można łatwo zidentyfikować za pomocą metod rankingowych, tak więc można powiedzieć, że transfer wiedzy dostępny jest na pewnym, dość abstrakcyjnym poziomie pomiędzy różnymi modelami danych.

Uniwersalne maszyny uczące nie są ograniczone do żadnej konkretnej metody, ale starają się wyodrębnić i dokonać transferu nowych cech do nowego algorytmu, zwiększając pulę potencjalnie użytecznych cech. Maszyny cech wsparcia (SFM) stanowią alternatywę dla podejścia SVM, wykorzystując liniowe funkcje dyskryminacyjne zdefiniowane w tak zwiększonych przestrzeniach [65]. Dla każdego wektora $\mathbf{X}$ istnieje n cech wejściowych plus m cech kernelowych $Z_i(\mathbf{X}) = K(\mathbf{X}, \mathbf{X}_i), i = 1..m$ zdefiniowanych dla każdego wektora treningowego. Modele liniowe w przestrzeni kernelowej są tak samo dokładne, jak kernelowy SVM, ale jawne stworzenie tej przestrzeni pozwala na większą elastyczność. Proste rozwiązania w oryginalnej przestrzeni cech $\mathbf{X}$ nie są pomijane, możliwy jest transfer informacji z innych modeli, mieszanie różnych rodzajów kerneli z różnymi parametrami, zapewniając w ten sposób wielorozdzielczość w różnych częściach przestrzeni. Jest to istotny krok w stronę meta-uczenia [49, 64, 16, 23].

6.5 Uniwersalne maszyny uczące

Każdy system ma jakąś szczególną specyficzność, która czyni go odpowiednim dla pewnej grupy problemów. Odkrycie tej specyficzności i znalezienie właściwego modelu nie jest łatwe. Realizuje się je zwykle poprzez eksperymenty w postaci kombinacji wstępnego przetwarzania danych, filtrowania i selekcji, klasteryzacji, klasyfikacji lub aproksymacji skombinowanych z procedurami metauczenia takimi jak stacking, boosting, komitety i inne. Liczba możliwych kombinacji różnych modułów w aplikacjach takich jak Weka [92] jest niezwykle duża (dla Weki $> 10^7$), a nowe składniki są wciąż dodawane przez ich twórców.

Pierwszym z rozwiązań jakie się nasuwa, jest znalezienie ogólnej charakterystyki danych i porównanie jej z charakterystyką danych referencyjnych, dla których został sporządzony ranking wyników różnych systemów [9]. Jednakże nie jest jasne jakie charakterystyki danych są niezbędne do istotnego porównania, a mała zmiana w rozkładzie danych może nieść ze sobą silny wpływ na rodzaj i kształt granic decyzji.

Drugie z rozwiązań to próba połączenia różnych transformacji danych w pewną strukturę, która systematycznie będzie zwiększać swoją złożoność. Takie rozwiązanie można sformułować jako generalną teorię uczenia w oparciu o transformacje, budowania modeli bazujących na złożeniu transformacji i ich kombinowaniu przy pomocy komitetów lokalnie kompetentnych modeli [17].

Sukces poszukiwania najlepszego zestawienia transformacji leży w zdolnościach transformacji do wydobywania użytecznych informacji, a co za tym idzie cech, które pozwolą

poradzić sobie z takimi problemami jak analiza obrazów, przetwarzanie tekstów, analiza sygnałów itd. Uniwersalne uczenie wydaje się być zatem nieuchwytnym snem, który nie może być zrealizowany bez różnorodnych transformacji, charakterystycznych dla określonego problemu. Znalezienie właściwej reprezentacji problemu to więcej niż połowa rozwiązania.

Nasuwa się zatem pytanie – jakie rodzaje cech są niezbędne dla różnych problemów? Sieci neuronowe bardzo często nazywane są uniwersalnymi aproksymatorami [43], jednakże ta właściwość okazuje się nie być wystarczającą do znalezienia interesujących rozwiązań. Po pierwsze, może być dosyć trudnym znaleźć dobre rozwiązania dla wysoce nieseparowalnych problemów [26, 38]. Po drugie, nawet jeśli zostanie zbudowany dobry model, to możliwe że istnieje znacznie prostsze, bardziej ogólne i zrozumiałe rozwiązanie. Może być ono znalezione przez tworzenie, agregację i selekcję użytecznych cech, odpowiadające rozszerzaniu i kurczeniu się przestrzeni, w której szukane są proste rozwiązania.

Formalnie rzecz ujmując, algorytm ULM składa się z dwóch modułów: konstruktorów cech i prostych klasyfikatorów. Uczenie maszynowe jest skoncentrowane na bezpośredniej analizie danych wejściowych, używając do tego miar odległości, liniowych kombinacji wartości poszczególnych cech itp., jednakże w wielu przypadkach takie transformacje przestrzeni cech nie są wystarczające.

Zestaw oryginalnych cech stanowi fragmentaryczny opis problemu. Pomijając ten fakt, część z nich może okazać się bezpośrednio użyteczna i taka powinna zostać wybrana i przetestowana. Inne cechy mogą zostać przetworzone przez rozmaite transformacje i filtry w celu wydobywania użytecznej informacji. Nowe cechy mogą być scharakteryzowane z punktu widzenia tego, jaki typ regionów decyzji dostarczają i jaka jest złożoność procesu szukania. Jednocześnie powinno się ustalić hierarchię cech pod względem ich złożoności.

Przeanalizowano wiele różnych typów cech oraz sposobów wydobywania użytecznych informacji z oryginalnych danych, począwszy od cech binarnych, przez kombinacje cech pochodzących z prostych drzew decyzji, zlokalizowanych klastrów po rzutach na prostą, po cechy bazujące na kernelach z różną dyspersją, a więc zapewniające adaptacyjną regularyzację. Nie wyczerpuje to oczywiście wszystkich możliwości konstrukcji informatywnych cech. Podczas gdy wiele wysiłku zostało poświęconego zagadnieniu selekcji cech [42], znacznie mniej wysiłku poświęcono konstrukcji cech [57].

Systematyczna eksploracja cech o rosnącej złożoności umożliwia odkrywanie prostych modeli, które bardziej wyszukane systemy uczenia mogą pominąć. Przykładowo znaczna część problemów używanych do porównań okazuje się być raczej trywialna i może być rozwiązana przy pomocy pojedynczej cechy binarnej, nominalnej, czy też nowej cechy skonstruowanej przez rzut na linię łączącą średnie dwóch klas. Cechy kernelowe oferują atrakcyjną alternatywę dla obecnych rozwiązań SVM bazujących na kernelach, oferując wielorozdzielczość i adaptacyjną regularyzację. Analiza obrazów, strumienie multimedialne czy biosekwencje mogą wymagać bardziej wyszukanych dróg konstrukcji cech bazujących na prostszych cechach.

Uniwersalne maszyny uczące opierają się na połączeniu zestawu różnych, opisywanych w niniejszej rozprawie metod: przetwarzania i generowania cech, ich późniejszej selekcji oraz klasyfikacji w nowo powstałej przestrzeni cech. Poszukiwane jest najlepsze zestawienie wspomnianych przed chwilą algorytmów, w celu znalezienia dobrego i jak najprostszego modelu dla konkretnego zbioru danych.

Ogólny schemat ULM przedstawiono na rysunku 6.3. Początkowo należy w nim określić, które z zaimplementowanych metod będą uwzględniane w kolejnych etapach

działania ULM, następnie bazując na wyborze dokonany przez użytkownika algorytm wygeneruje nową przestrzeń cech, w której po wcześniejszym ograniczeniu jej poprzez jedną z wybranych metod selekcji cech znajdzie najlepsze rozwiązanie używając do tego określonego klasyfikatora. Poniżej wymieniono (podzielone na kategorie) wszystkie algorytmy wchodzące w skład obecnej implementacji algorytmu ULM.

1. Wstępne przetwarzanie danych:

- standaryzacja,
- normalizacja długości wektorów.

2. Generowanie nowych cech:

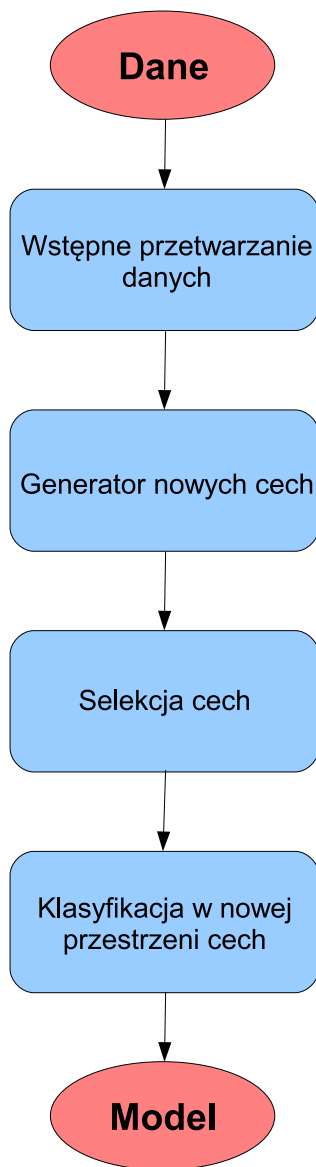
- dane oryginalne,
- n składników głównych (PCA),
- n cech pochodzących ze skalowania wielowymiarowego (MDS),
- cechy uzyskane za pomocą analizy dyskryminacyjnej Fishera (FDA),
- rzuty na n przypadkowych kierunków,
- przedziały (czyste klastry) wyodrębnione z n przypadkowych kierunków (aRPM),
- rzuty (i ich przedziały) na kierunki wyznaczone przez centra klas,
- rzuty (i ich przedziały) na kierunki wyznaczone przez centra n klastrów,
- przekształcenie triangulacyjne (TriVis),
- mapowanie SVMowe,
- gaussowskie cechy kernelowe o różnej dyspersji,
- lokalnie zoptymalizowane gaussowskie cechy kernelowe (LOK),
- n cech wyodrębnionych z drzewa decyzji.

3. Selekcja atrybutów:

- współczynnik korelacji,
- informacja wzajemna,
- współczynnik Fishera,

4. Klasyfikacja w nowej przestrzeni cech:

- klasyfikator naiwnego Bayesa (NB),
- metoda k najbliższych sąsiadów (kNN),
- drzewo decyzji (C4.5),
- dyskryminacja liniowa (LDA),
- maszyny wektorów wspierających (SVM).



Rysunek 6.3: Schemat działania ULM.

6.6 Wyniki uzyskane przez algorytm ULM

W tym podrozdziale przedstawione zostały rezultaty działania algorytmu ULM. Metoda ta została porównana z innymi popularnymi klasyfikatorami opisanymi w rozdziale 5. Do analizy posłużono się jedenastoma zbiorami pochodzącymi z repozytorium UCI [3], których krótką charakterystykę można obejrzeć w dodatku A. Wybór danych pochodzących z repozytorium UCI podyktowany był powszechną dostępnością źródła danych, co pozwala na porównanie rezultatów z wynikami otrzymanymi przez innych badaczy.

Proces testowania polegał na dziesięć razy powtórzonej 10-krotnej krosvalidacji, a wyniki w postaci średnich poprawności klasyfikacji i odchyłeń standardowych

zestawiono w tabeli 6.2. Należy tutaj zaznaczyć, że parametry konfiguracyjne wszystkich porównywanych klasyfikatorów były dobierane przy użyciu wewnętrznej krosvalidacji (na części treningowej), a zamieszczone wyniki innych metod uzyskano wykorzystując do obliczeń pakiety Weka [92], GhostMiner [28] oraz Intemi [47].

Dane	SSV	kNN	NB	SVM(L)	SVM(G)	ULM
Breast-cancer	76.9 ± 5.4	73.6 ± 7.1	73.8 ± 7.9	73.2 ± 6.1	75.6 ± 5.3	76.9 ± 6.4
Breast-w	95.8 ± 2.2	96.5 ± 2.2	96.2 ± 2.3	96.6 ± 2.0	96.7 ± 1.8	97.2 ± 2.3
Credit-a	85.6 ± 4.4	83.0 ± 4.3	77.0 ± 4.5	86.3 ± 2.8	86.2 ± 2.9	86.4 ± 3.3
Credit-g	70.2 ± 3.4	73.5 ± 3.2	75.2 ± 3.8	73.9 ± 4.6	74.7 ± 4.0	76.1 ± 4.7
Diabetes	73.5 ± 4.8	74.9 ± 4.8	75.3 ± 4.3	76.8 ± 4.9	76.4 ± 4.2	77.1 ± 4.0
Heart-c	78.7 ± 6.9	82.9 ± 6.3	82.4 ± 6.7	82.6 ± 6.3	80.6 ± 7.9	84.0 ± 5.9
Heart-statlog	80.8 ± 7.7	83.5 ± 6.2	83.9 ± 7.3	83.4 ± 7.1	83.4 ± 6.5	84.5 ± 7.0
Hepatitis	83.6 ± 11.7	85.6 ± 11.3	91.2 ± 9.1	83.2 ± 11.5	84.8 ± 11.9	89.6 ± 10.7
Ionosphere	87.2 ± 5.2	86.3 ± 4.9	84.2 ± 6.1	87.7 ± 4.6	94.6 ± 3.6	94.4 ± 3.9
Liver-disorders	67.4 ± 6.9	63.9 ± 7.4	56.2 ± 7.9	68.4 ± 7.3	70.3 ± 7.9	72.2 ± 6.9
Vote	96.9 ± 3.8	92.7 ± 5.5	91.9 ± 4.9	96.1 ± 3.8	96.8 ± 3.1	96.9 ± 3.4

Tabela 6.2: Wyniki poprawności klasyfikacji.

Analizując wyniki przedstawione w tabeli 6.2 widać, że dla większości badanych zbiorów (krótki opis zbiorów w dodatku A), a więc: *Breast-cancer*, *Breast-w*, *Credit-a*, *Credit-g*, *Diabetes*, *Heart-c*, *Heart-statlog*, *Liver-disorders* oraz *Vote* algorytm ULM osiąga najlepsze wyniki średnich dokładności klasyfikacji. Jedyne w przypadku zbioru *Hepatitis* wynik uzyskany przez klasyfikator Naiwnego Bayesa oferuje poprawność lepszą o 1.5%, aczkolwiek przy odchyleniu standardowym na poziomie 10% różnica ta nie jest znacząca. Podobnie w przypadku zbioru *Ionosphere* rezultat jest praktycznie rzecz biorąc niemal identyczny jak ten uzyskany przez SVM z kernelem gaussowskim (różnica wynosi 0.2% przy 4% odchylenia standardowego).

Przedstawione wyniki pozwalają wysunąć przypuszczenie, że otrzymane średnie dokładności klasyfikacji są limitem tego, co można “wyciągnąć” z przetestowanych zbiorów danych, ponieważ najczęściej nie jeden, a kilka klasyfikatorów oferuje dokładności zbliżone do najlepszego otrzymanego rezultatu.

Rozkład procentowy rodzajów cech wybieranych przez algorytm ULM w trakcie testów na w/w danych został pominięty, gdyż dla każdego zbioru jest on inny (różni się on także w poszczególnych foldach procesu krosvalidacji). Oznacza to, że dla każdego zadania inny rodzaj cech okazuje się być bardziej istotny.

Warto zaznaczyć, że praktycznie dla każdego przeanalizowanego zbioru danych do końcowej klasyfikacji wystarczającym okazało się być użycie liniowej dyskryminacji z maksymalizacją marginesu, a zastosowanie innych klasyfikatorów nie przynosiło poprawy rezultatów. Oczywiście nie oznacza to, że przy innych jakichś trudnych, specyficznych danych użycie innego algorytmu nie przyniesie poprawy wyników.

Przez długi okres czasu, opierając się m.in. na wynikach uzyskanych w projekcie Statlog [67] uznawano, że nie ma takiego klasyfikatora, który byłby najlepszy dla dowolnego problemu. Zaprezentowany w tej pracy algorytm ULM i otrzymane rezultaty pozwalają niejako przeformułować to twierdzenie, mówiąc że przy użyciu odpowiedniej reprezentacji danego problemu (analizując go we właściwie określonej przestrzeni cech) nawet prosty klasyfikator liniowy może rozwiązać dowolny problem w sposób bliski optymalnemu.

Rozdział 7

Podsumowanie

Najważniejszym elementem uczenia maszynowego, inteligencji obliczeniowej, rozpoznawania wzorców i dziedzin pokrewnych jest tworzenie inteligentnych algorytmów, które potrafią się automatycznie konfigurować i prowadzą do odkrycia dobrych i prostych modeli dla dowolnych zbiorów danych.

Niniejsza rozprawa skupia się w głównej mierze na generowaniu transformacji, kategoryzacji różnych typów cech, tworzeniu nowych cech, uczeniu się z innych modeli poprzez transfer wiedzy, zrozumieniu jak wygląda rozkład danych w rozszerzonej przestrzeni cech oraz znajdowaniu optymalnych klasyfikatorów działających w takiej przestrzeni.

Systematyczne poszukiwanie modeli opartych na cechach nowego rodzaju umożliwia odkrycie prostych modeli, które bardziej zaawansowane systemy uczenia pomijają. Opisane konstruktory cech daleko wykraczają poza liniowe kombinacje oferowane m.in. przez algorytmy PCA czy ICA. W szczególności cechy oparte na kernelach stanowią atrakcyjną alternatywę dla popularnie stosowanego algorytmu SVM, oferując możliwość wielorozdzielczości i adaptacyjnej regularyzacji. Zaprezentowanych zostało kilka nowych metod konstruowania cech, a ich użyteczność została dokładnie przeanalizowana. Mieszanie różnych kerneli i używanie różnych typów cech zapewnia znacznie większą elastyczność tworzenia granic decyzji. Dodawanie specyficznych cech wsparcia ułatwia odkrywanie wiedzy tkwiącej w danych. Dobra generalizacja jest uzyskiwana dzięki dużym, czystym klastrom wektorów, które mogą zostać pozyskane przy pomocy specyficznych filtrów. Konstruowanie różnych filtrów informacji prowadzi do interesujących spojrzeń na dane, odkrywając nieliniowe struktury w danych, które (jeżeli zostaną zauważone) mogą być łatwo poddane obróbce przez konkretne transformacje. Systemy, które aktywnie próbują dane, próbując “zobaczyć je” poprzez filtry informacji, wydają się być znacznie bardziej elastyczne niż klasyfikatory pracujące w niezmiennych przestrzeniach wejściowych.

Chociaż ostatecznym celem uczenia jest odkrycie ciekawych modeli danych, należy zwrócić większą uwagę na reprezentacje pośrednie, czyli obraz danych po transformacji. Poszukiwanie interesujących spojrzeń na dane przez konstruowanie interesujących filtrów stanowi najlepszą drogę do budowy praktycznych aplikacji, które automatycznie zbudują interesujące modele dla dowolnego zbioru danych. Obiekty mogą mieć różnorodną i skomplikowaną strukturę, a różne kategorie mogą być identyfikowane w różnych przestrzeniach cech uzyskanych za pomocą takich filtrów i transformacji. Kiedy struktura danych w rozszerzonej przestrzeni będzie rozpoznana, wówczas proces uczenia staje się znacznie uproszczony przez zmianę celu uczenia na łatwiejszy.

Neurokognitywna informatyka czerpie inspiracje z procesów neurobiologicznych

odpowiedzialnych za uczenie się i tworzy dobrą podstawę dla idei uczenia przedstawionej w niniejszej rozprawie. Do tej pory w inteligencji obliczeniowej wykorzystano tylko kilka ogólnych inspiracji, jak na przykład neurony progowe łączone w sieci, które realizują równoległe przetwarzanie rozproszone. Nawet przy naszym ograniczonym rozumieniu działania mózgu wiele innych inspiracji może być zaczerpnięte i wykorzystywane w praktycznym uczeniu i algorytmach rozpoznawania obiektów. Równoległe strumienie dodatkowych informacji, zorganizowane w hierarchiczny sposób [41] można odnieść do wielu filtrów informacji, generujących nowe cechy wyższego rzędu.

Uzyskane wyniki klasyfikacji rzeczywistych danych wskazują na duże możliwości analizy w oparciu o metody generowania nowych, użytecznych cech. Już rezultaty uzyskane na pojedynczych typach cech sugerują trafność zaproponowanego podejścia, a dodatkowa integracja różnych rodzajów atrybutów wykazała, iż w większości wypadków uzyskane rezultaty były porównywalne lub nawet przewyższały wyniki uzyskane powszechnie uznanymi metodami.

Poniżej zestawiono opracowane przeze mnie metody wraz z ich krótkim podsumowaniem. Należy tutaj zaznaczyć, iż wszystkie zaprezentowane metody generacji nowych, użytecznych cech są składową jednego algorytmu - Uniwersalnych Maszyn Uczących. Użyteczność każdego rodzaju atrybutów starano się oceniać niezależnie, co doprowadziło do powstania kilku niezależnych metod klasyfikacji opisanych w niniejszej pracy.

Algorytm TriVis [66], opisany w podrozdziale 4.4, tworzy reprezentację oryginalnych danych w dwuwymiarowej przestrzeni, przy jednoczesnym zachowaniu tyłu odległości pomiędzy wektorami danych, ile jest tylko możliwe. Otrzymana dwuwymiarowa reprezentacja może zostać wykorzystana do zwiększenia puli potencjalnie użytecznych cech. Ponadto, algorytm może zostać wykorzystany do inicjalizacji metody MDS. Zabieg ten powoduje zmniejszenie początkowego stresu i co za tym idzie, znaczne ograniczenie liczby iteracji.

Algorytm mapowania SVMowego [63], zaprezentowany w podrozdziale 4.5, służy do redukcji oryginalnej przestrzeni cech przy pomocy SVMa z kernelem liniowym. Często reprezentacja w jednym wymiarze okazuje się być wystarczająca i taka może zostać użyta do wykreślenia prawdopodobieństw warunkowych. Dalsze zwiększanie liczby wymiarów polega na generowaniu kolejnych kierunków dyskryminujących, co zostało dokładniej opisane w opisie algorytmu. Podobnie jak w przypadku wizualizacji triangularyzacyjnej, przy redukcji oryginalnych danych do 2 lub 3 wymiarów, metody tej można użyć do przedstawienia problemu w postaci graficznej.

Algorytm SFM [65], opisany w podrozdziale 5.8, bazuje na generowaniu nowych "cech wspierających" przy pomocy różnych kerneli, przypadkowych liniowych rzutów oraz ograniczonych rzutów, które poprzedzają selekcję cech i liniową dyskryminację (jeśli nowe cechy mają być wykorzystane do budowy odrębnego klasyfikatora). W obecnej wersji algorytm używa trzech typów cech, które zostały przedstawione przy opisie algorytmu. Niezaprzeczalnymi zaletami opisywanego algorytmu jest oferowana przez niego wielorozdzielczość, jak również jawna reprezentacja i systematyczne możliwości rozszerzania przestrzeni cech, podczas gdy wielokernelowe modele są mało wykorzystywane i trudne w użyciu.

Algorytm aRPM [15], zaprezentowany w podrozdziale 5.9, został zainspirowany przez mechanizmy biologiczne zachodzące w mózgach. Generuje on wystarczającą (do poprawnej klasyfikacji) liczbę przypadkowych kierunków, a następnie po zrzutowaniu na nie oryginalnych danych, wybiera te najbardziej przydatne, tworząc użyteczne

cechy (ograniczone projekcje, zawierające przypadki tylko z jednej klasy). W celu sprawdzenia użyteczności generowanych w ten sposób cech, utworzono klasyfikator, który wykorzystuje mechanizm WTA. Wagi w tym algorytmie są wybierane losowo, zaś ich wartości nie zmieniają się. Tak więc nie potrzeba uczenia się wag (w przeciwieństwie do wszystkich innych metod uczenia sieci neuronowych). Zaskakującym jest, że wyniki modelu, który nie przeprowadza żadnej optymalizacji są dla popularnych zbiorów danych co najmniej tak dobre, jak te uzyskane przez sieci MLP. Dodatkowo metoda ta idealnie radzi sobie z problemami o złożonej logice. Ponadto algorytm aRPM umożliwia łatwą ocenę zaufania otrzymanych rezultatów.

Algorytm aRPM z maksymalizacją marginesu [60], opisany w podrozdziale 5.10, jest modyfikacją podstawowego algorytmu aRPM. W tej wersji największą uwagę kładzie się na maksymalizację marginesu klasyfikacji. Nowe projekcje dodawane są tylko wtedy, jeżeli zwiększają poprawność klasyfikacji tych przypadków, które znajdują się albo po złej stronie, albo w pobliżu granicy decyzyjnej. Obecnie metoda używa dwóch rodzajów atrybutów - cech wygenerowanych przy pomocy rzutów na losowe kierunki, oraz cech opartych o gaussowskie kernele. Cechy te są następnie odfiltrowywane przez indeks oceniający szerokość marginesu klasyfikacji. Tak jak w poprzedniej wersji algorytmu, tak i tutaj po rzutowaniu danych na losowe kierunki, poszukiwane są przedziały (klastry), w których występują wektory tylko z jednej klasy. Algorytm inkrementacji rozszerza przestrzeń cech aż do momentu, kiedy nie uzyskuje się wzrostu dokładności klasyfikacji. Do określenia ostatecznej decyzji algorytm aRPM z maksymalizacją marginesu używa mechanizmu WTA albo liniowej dyskryminacji.

Algorytm LOK [61], zaprezentowany w podrozdziale 6.3, opiera się na generacji nowych, użytecznych, ograniczonych cech kernelowych. Pokazano, że LOK jest całkiem dobrą i prostą metodą generowania takich cech, które mogą zostać użyte do budowy lepszej kernelowej przestrzeni cech w algorytmie SVM. W przypadku gaussowskich cech kernelowych, nie ma powodu dlaczego używać takiej samej dyspersji dla wszystkich wektorów wsparcia. Te wektory wsparcia, które są daleko od granicy decyzji po złej stronie, powinny dostarczyć cech kernelowych, które wykluczać będą lokalne sąsiedztwo, podczas kiedy takie same wektory wsparcia, ale znajdujące się po właściwej stronie powinny używać raczej dużych dyspersji. Wektory wsparcia blisko granic decyzji powinny cechować się raczej małymi dyspersjami. Algorytm LOK tworzy takie lokalnie optymalizowane kernele gaussowskie. Aby wykorzystać algorytm LOK w charakterze klasyfikatora i w ten sposób wykazać istotność i jakość generowanych nowych atrybutów, do znalezienia rozwiązania w nowej przestrzeni cech używany jest mechanizm WTA lub liniowa dyskryminacja.

Algorytm ULM [30], opisany w podrozdziale 6.5, opiera się na połączeniu zestawu różnych, opisywanych w niniejszej rozprawie metod: przetwarzania i generowania cech, ich późniejszej selekcji oraz klasyfikacji w nowo powstałej przestrzeni cech. Do tworzenia nowych atrybutów wykorzystywane są zarówno metody przedstawione powyżej, jak i mechanizmy transferu wiedzy oraz metauczenia. ULM poszukuje najlepszego zestawienia wspomnianych przed chwilą algorytmów, w celu znalezienia dobrego i jak najprostszego modelu dla określonego zbioru danych.

Warto wspomnieć także o opisanym w podrozdziale 5.4.1 usprawnieniu algorytmu drzew decyzji, polegającym na modyfikacji kryterium podziału przez zmianę entropii Shannona na entropię Renyiego lub Tsallisa [62]. Dzięki temu algorytm może łatwo poradzić sobie z poprawną klasyfikacją jednej lub wielu mało licznych klas. Dodatkowy parametr zapewnia możliwość zwiększenia poprawności klasyfikacji klas o różnej

liczebności wektorów, co sprawia że zmodyfikowany algorytm drzew decyzji staje się bardziej atrakcyjnym w porównaniu do oryginalnej wersji opartej o entropię Shannona. Tak usprawnione drzewo tworzyć będzie znacznie bardziej ciekawe reguły, które wykorzystywane będą przez algorytm ULM w procesie transferu wiedzy.

Dodatek A

Charakterystyka zbiorów danych użytych w testach

Poniżej, w tabeli A.1 przedstawiono krótkie podsumowanie zbiorów danych używanych w niniejszej rozprawie doktorskiej. Dla każdego ze zbiorów wektory z wartościami brakującymi dla atrybutów zostały usunięte (jeżeli takowe występowały). Dodatkowo przy każdym ze zbiorów zaznaczono skąd został pobrany.

Dane	Ilość wektorów	Ilość cech	Ilość klas	Źródło pochodzenia
Appendicitis	106	7	2	UCI [3]
Arrhythmia	63	279	11	UCI [3]
Autos	159	25	6	UCI [3]
Balance-scale	625	4	3	UCI [3]
Breast-cancer	277	9	2	UCI [3]
Breast-w	683	9	2	UCI [3]
Car	1728	6	4	UCI [3]
Cmc	1473	9	3	UCI [3]
Colon-cancer	62	2000	2	Alon at all (1999) [2]
Credit-a	653	15	2	UCI [3]
Credit-g	1000	20	2	UCI [3]
Cylinder-bands	277	39	2	UCI [3]
Dermatology	358	34	6	UCI [3]
Diabetes	768	8	2	UCI [3]
DLBCL	47	4026	2	Alizadeh at all (2000) [1]
Ecoli	336	7	8	UCI [3]
Glass	214	9	6	UCI [3]
Haberman	306	3	2	UCI [3]
Heart-c	296	13	2	UCI [3]
Heart-statlog	270	13	2	UCI [3]
Hepatitis	80	19	2	UCI [3]
Ionosphere	351	34	2	UCI [3]
Iris	150	4	3	UCI [3]
Kr-vs-kp	3196	36	2	UCI [3]
Leukemia	72	7129	2	Golub at all (1999) [36]
Liver-disorders	345	6	2	UCI [3]
Lymph	148	18	4	UCI [3]
Parity8	256	8	2	zbiór sztuczny
Parity10	1024	10	2	zbiór sztuczny
Sonar	208	60	2	UCI [3]
Vote	232	16	2	UCI [3]
Vowel	990	13	11	UCI [3]
Wine	178	13	3	UCI [3]
Zoo	101	17	7	UCI [3]

Tabela A.1: Podsumowanie zbiorów danych.

Bibliografia

- [1] A.A. Alizadeh. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.
- [2] U. Alon. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:745–750, 1999.
- [3] A. Asuncion, D.J. Newman. UCI machine learning repository. <http://www.ics.uci.edu/mlearn/MLRepository.html>, 2007.
- [4] V. Barnett, T. Lewis. *Outliers in statistical data*. J. Wiley and Sons, 1978.
- [5] Y. Bengio, O. Delalleau, N. Le Roux. The curse of dimensionality for local kernel machines. Technical Report 1258, Département d’informatique et recherche opérationnelle, Université de Montréal, 2005.
- [6] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.
- [7] I. Borg, P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics, 2005.
- [8] B.E. Boser, I.M. Guyon, V.N. Vapnik. A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on COLT*, str. 144-152, ACM Press, Pittsburgh, PA, 1992.
- [9] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta. *Metalearning: Applications to Data Mining*. Cognitive Technologies. Springer, 2009.
- [10] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.
- [11] Yong-Qing Cheng, Yong-Ming Zhuang, Jing-Yu Yang. Optimal Fisher discriminant analysis using the rank decomposition. *Pattern Recognition*, 25(1):101–111, 1992.
- [12] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.
- [13] T.F. Cox, M.A.A. Cox. *Multidimensional Scaling*, 2nd Ed. Chapman and Hall, 2001.
- [14] J. Doak. *Intrusion detection: the application of feature selection, a comparison of algorithms and the application of a wide area network analyzer*. Praca magisterska, University of California, Davis Dept. of Computer Science, 1992.

- [15] W. Duch, T. Maszczyk. Almost random projection machine. *Lecture Notes in Computer Science*, 5768:789–798, 2009.
- [16] W. Duch, T. Maszczyk, M. Grochowski. Optimal Support Features for Meta-learning. *Meta-learning in Computational Intelligence. Studies in Computational Intelligence*, str. 317–358, Springer, 2011.
- [17] W. Duch, L. Irt. Committees of undemocratic competent models. *Proc. of Int. Conf. on Artificial Neural Networks (ICANN)*, Istanbul, str. 33–36, 2003.
- [18] W. Duch, R. Adamczak, Y. Hayashi. Eliminators and classifiers. In *7th International Conference on Neural Information Processing (ICONIP)*, str. 1029–1034. Dae-jong, Korea, 2000.
- [19] W. Duch, N. Jankowski. Transfer functions: hidden possibilities for better neural networks. In *9th European Symposium on Artificial Neural Networks*, str. 81–94, Brusells, Belgium, 2001.
- [20] W. Duch. Filter methods. In *Feature extraction, foundations and applications*, str. 89–118. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2006.
- [21] W. Duch, R. Setiono, J. Zurada. Computational intelligence methods for understanding of data. *Proceedings of the IEEE*, 92(5):771–805, 2004.
- [22] W. Duch. Neurocognitive Informatics Manifesto. *Series of Information and Management Sciences*, str. 264–282, Kunming-Banna, Yunan, China, 2009.
- [23] W. Duch, K. Grudziński. Meta-learning: searching in the model space. *Proceedings of the International Conference on Neural Information Processing*, str. 235–240, Shanghai, 2001.
- [24] W. Duch. Quantitative measures for the self-organized topographical mapping. *Open Systems and Information Dynamics*, 2:295–302, 1995.
- [25] W. Duch, N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2:163–213, 1999.
- [26] W. Duch. k-separability. *Lecture Notes in Computer Science*, 4131:188–197, 2006.
- [27] W. Duch, R. Adamczak, K. Grąbczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.
- [28] W. Duch, N. Jankowski, K. Grąbczewski, A. Naud, R. Adamczak. Ghostminer data mining software. Technical report, Department of Informatics, Nicolaus Copernicus University, 2000–2008.
- [29] W. Duch, N. Jankowski, T. Maszczyk. Make it cheap: learning with $O(nd)$ complexity. In *World Congress on Computational Intelligence*, IEEE Press, 2010. (w druku)
- [30] W. Duch, T. Maszczyk. Universal learning machines. *Lecture Notes in Computer Science*, 5864:206–215, 2009.

- [31] R.O. Duda, P.E. Hart, D.G. Stork. *Patter Classification*. J. Wiley and Sons, New York, 2001.
- [32] G.B. Huang, L. Chen, C.K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17:879-892, 2006.
- [33] S.E. Fahlman. Faster-learning variations on back-propagation: an empirical study. *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, San Francisco, 1989.
- [34] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. *From Data Mining to Knowledge Discovery in Databases*, 1996.
- [35] J.H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.
- [36] T.R. Golub. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [37] R.L. Gorsuch. *Factor Analysis*. Erlbaum, Hillsdale, NJ, 1983.
- [38] M. Grochowski, W. Duch. Constructive Neural Network Algorithms that Solve Highly Non-Separable Problems. *Studies in Computational Intelligence*, 258:49–70, Springer, 2010.
- [39] K. Grąbczewski, W. Duch. The separability of split value criterion. In *Proceedings of the 5th Conf. on Neural Networks and Soft Computing*, str. 201–208, Zakopane, Poland, 2000.
- [40] M. Grochowski, W. Duch. Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. *Lecture Notes in Computer Science*, 4668:180–189, 2007.
- [41] S. Grossberg. The complementary brain: Unifying brain dynamics and modularity. *Trends in Cognitive Sciences*, 4:233–246, 2000.
- [42] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh. *Feature extraction, foundations and applications*. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2006.
- [43] L. Györfi, M. Kohler, A. Krzyżak, H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, 2002.
- [44] S. Haykin. *Neural Networks - A Comprehensive Foundation*. Maxwell MacMillan Int., New York, 1994.
- [45] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.
- [46] F. Jäkel, B. Schölkopf, F.A. Wichmann. Does cognitive science need kernels? *Trends in Cognitive Sciences*, 13(9):381–388, 2009.

- [47] N. Jankowski, K. Grąbczewski. Universal Meta-learning Architecture and Algorithms. *Meta-learning in Computational Intelligence. Studies in Computational Intelligence*, 358:1-76, 2011.
- [48] N. Jankowski, W. Duch. Optimal transfer function neural networks. In *9th European Symposium on Artificial Neural Networks*, str. 101–106, Bruges, Belgium, 2001.
- [49] N. Jankowski. *Meta-uczenie w inteligencji obliczeniowej*. Exit, Warszawa, 2011.
- [50] C. Jones, R. Sibson. What is projection pursuit. *Journal of the Royal Statistical Society A*, 150:1–36, 1987.
- [51] A. Kalousis, J. Prados, M. Hilario. Stability of feature selection algorithms: a study on high dimensional spaces. *Knowledge and Information Systems*, 12:95–116, 2007.
- [52] R. Kohavi, G. John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97:273-324, 1997.
- [53] J. Korbicz, A. Obuchowicz, D. Uciński. *Sztuczne sieci neuronowe. Podstawy i zastosowania*. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1994.
- [54] J. Koronacki, J. Ćwik. *Statystyczne systemy uczące się*. Wydawnictwa Naukowo-Techniczne, 2005.
- [55] J. A. Lee, M Verleysen. *Nonlinear dimensionality reduction*. Information Science and Statistics, Springer, 2007.
- [56] H. Leung, S. Haykin. Detection and estimation using an adaptive rational function filters. *IEEE Transactions on Signal Processing*, 12:3365–3376, 1994.
- [57] Huan Liu, Hiroshi Motoda. *Feature extraction, construction and selection: a data mining perspective*, SECS 453. Kluwer Academic, Boston, 1998.
- [58] W. Maass, T. Natschläger, H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14:2531–2560, 2002.
- [59] W. Maass, H. Markram. Theory of the computational function of microcircuit dynamics. In *Microcircuits. The Interface between Neurons and Global Brain Function*, str. 371–392, MIT Press, 2006.
- [60] T. Maszczyk, W. Duch. Almost random projection machine with margin maximization and kernel features. *Lecture Notes in Computer Science*, 6353:40–48, 2010.
- [61] T. Maszczyk, W. Duch. Locally Optimized Kernels. *Lecture Notes in Computer Science*, 2012. (w druku)
- [62] T. Maszczyk, W. Duch. Comparison of shannon, renyi and tsallis entropy used in decision trees. *Lecture Notes in Computer Science*, 5097:643–651, 2008.
- [63] T. Maszczyk, W. Duch. Support vector machines for visualization and dimensionality reduction. *Lecture Notes in Computer Science*, 5163:346–356, 2008.

- [64] T. Maszczyk, M. Grochowski, W. Duch. Discovering Data Structures using Meta-learning, Visualization and Constructive Neural Networks. *Advances in Machine Learning II*, 262:467–484. Springer Series: Studies in Computational Intelligence, 2010.
- [65] T. Maszczyk, W. Duch. Support feature machines: Support vectors are not enough. In *World Congress on Computational Intelligence*, str. 3852–3859, IEEE Press, 2010.
- [66] T. Maszczyk, W. Duch. Triangular Visualization. *Lecture Notes in Computer Science*, 6113:445-452, 2010.
- [67] D. Michie, D.J. Spiegelhalter, C.C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, London, 1994.
- [68] A. Newell. *Unified theories of cognition*. Harvard Univ. Press, Cambridge, MA, 1990.
- [69] R.C. O'Reilly, Y. Munakata. *Computational Explorations in Cognitive Neuroscience*. MIT-Press, 2000.
- [70] J.R. Quinlan. *C 4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [71] L. Rokach, O. Maimon. *Data Mining with Decision Trees: Theory and Applications*, World Scientific Publishing Company, 2007.
- [72] A. Renyi. *Probability Theory*. North-Holland, Amsterdam, 1970.
- [73] D.E. Rumelhart, G.E. Hinton, R.J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the microstructure of cognition*. MIT Press, Massachusetts, 1986.
- [74] L. Rutkowski. *Flexible Neuro-Fuzzy Systems*. Kluwer Academic, 2004.
- [75] L. Rutkowski. *Metody i techniki sztucznej inteligencji*. Wydawnictwo Naukowe PWN, 2006.
- [76] M.E. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.
- [77] W. Duch, R. Adamczak, G.H.F. Diercksen. Classification, association and pattern completion using neural similarity based methods. *Applied Mathematics and Computer Science*, 10:101–120, 2000.
- [78] J.L. Schafer, M.K. Olsen. Multiple imputation for multivariate missing-data problems: a data analyst's perspective. *Multivariate Behavioral Research*, 33:545-571, 1998.
- [79] C.E. Shannon, W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Ill., 1964.
- [80] B. Schölkopf, A.J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.

- [81] F. Schwenker, H.A. Kestler, G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14:439–458, 2001.
- [82] P.L. Smith, R. Ratcliff. Psychology and neurobiology of simple decisions. *Trends in Neurosciences*, 27:161–168, 2004.
- [83] P. Somol, J. Novovičová, P. Pudil. Efficient Feature Subset Selection and Subset Size Optimization. *Pattern Recognition Recent Advances, INTECH*, str. 75–97, 2010.
- [84] S. Sonnenburg, G. Raetsch, C. Schaefer, B. Schoelkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.
- [85] R. Tadeusiewicz. *Sieci neuronowe*. Akademicka Oficyna Wydawnicza RM, 1993.
- [86] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.
- [87] C. Tsallis, R.S. Mendes, A.R. Plastino. The role of constraints within generalized nonextensive statistics. *Physica*, 261A:534–554, 1998.
- [88] I.W. Tsang, J.T. Kwok, Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.
- [89] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.
- [90] S. Ullman. *High-level vision: Object recognition and visual cognition*. MIT Press, Cambridge, MA, 1996.
- [91] A.R. Webb. *Statistical Pattern Recognition*. J. Wiley and Sons, 2002.
- [92] I.H. Witten, E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
- [93] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.