Instytut Podstawowych Problemów Techniki
Polska Akademia Nauk

Warszawa, ul. Świętokrzyska 21

PRACA DOKTORSKA

TEMAT: **Konstrukcja klasyfikatora obiektów z wykorzystaniem algorytmu badania rozdzielności liniowej dwóch zbiorów**

AUTOR: mgr inż. Dorota Daniecka

PRACA WYKONANA POD KIERUNKIEM: prof. dr hab. Witold Kosiński

STRESZCZENIE: W pracy przedstawiono różne definicje rozdzielności dwóch zbiorów. Zamieszczono szkic algorytmu i sam algorytm, który umożliwia konstrukcję hierarchicznego klasyfikatora binarnego. Algorytm ten w znaczącym stopniu — i jak pokazano w naturalny sposób — wykorzystuje algorytm badania ścisłej rozdzielności dwóch zbiorów.

Warszawa październik 2005

Pamięci B. B. Gaweł i K. J. Danieckich

Pragnę wyrazić szczególną wdzięczność prof. Kosińskiemu za każdy przecinek, który dostawił lub usunął, za orkę na ugorze orzeczeń, dopełnień i przydawek dr. Adamowi Jóźwikowi za algorytmiczną „perełkę" prof. Paszkowskiemu za „szczególny przypadek twierdzenia...", czyli tożsamość Schweinsa prof. Marksowi za każdy uśmiech i życzliwość, którymi obdarza Doktorantów moim Rodzicom za miłość i cudowną obecność moim Teściom za mojego Męża a mojemu Mężowi za to, że jest Aniołem Alicji Bednarskiej za Oddech, po którym wszystko stało się inne oraz wszystkim spotkanym do tej pory ludziom, którzy swoją postawą lub samym istnieniem czegoś mnie nauczyli i... Naturze za Piękno, jakim jest każdy nowy dzień.

Wprowadzenie

Być albo nie być — oto jest pytanie [...][1] to nie tylko jeden z nasłynniejszych cytatów literatury światowej, ale również jedno z postawowych pytań egzystencjalnych, które — biorąc pod uwagę trendy światowe — przybiera w dzisiejszych czasach często formę tytułu równie słynnej pozycji książkowej: *Mieć czy być*[2]? Jakbyśmy jednak nie zadawali tego pytania, próba odpowiedzi na nie oznacza opowiedzenie się za jedną z dwóch możliwości. Na uwspółcześnioną wersję przytoczonego pytania możemy więc spojrzeć jako na mechanizm jednego z najprostszych i najtańszych klasyfikatorów, który „nie bez błędów" pozwala szybko stwierdzić z jakim człowiekiem mamy do czynienia.

Klasyfikowanie, przyporządkowywanie do grupy pojęć — zjawisk, zachowań czy przedmiotów o określonych cechach — nie ma swego ściśle określonego początku historycznego i jest równie stare jak obecność człowieka na Ziemi. Znany jest natomiast początek i kierunki badań naukowych związanych z automatycznym klasyfikowaniem *obiektów* przy użyciu komputerów, a ściślej związanych z projektowaniem programów komputerowych implementujących algorytmy klasyfikujące. Używając terminu *obiekt*, mamy na myśli istotę tego, co stanowi przedmiot procesu klasyfikacji.

Konstrukcja programów rozpoznających opiera się na znajomości zbioru obiektów już sklasyfikowanych, tzn. zbioru uczącego. Możliwość uzyskiwania ogromnej ilości danych w ramach zbiorów uczących sprawia, że tworzenie i rozwijanie istniejących klasyfikatorów dwudecyzyjnych jest nie tylko użyteczne, ale pozwala również wykorzystać klasyfikatory tego typu jako podstawę do konstrukcji klasyfikatorów wielodecyzyjnych. Jakość klasyfikatorów zależy w największym stopniu od dwóch podstawowych czynników: jakości zbioru uczącego i algorytmu rozdzielającego. W pracy skupiono się na drugim z wymienionych elementów.

Celem rozprawy jest konstrukcja hierarchicznego klasyfikatora liniowego[3], wykorzystującego algorytm optymalnego rozdzielania dwóch, skończonych zbiorów punktów reprezentujących dwie klasy obiektów ze zbioru uczącego. Każdy z obiektów reprezentowany jest przez parę: n–wymiarowy wektor cech i etykietę, która określa przynależność obiektu do jednej z dwóch klas. Ze względu na to, że wektor cech jest jednocześnie punktem w Euklidesowej przestrzeni E^n, zbiór uczący obiektów możemy identyfikować ze zbiorem punktów lub wektorów wiodących w przestrzeni cech. Takie potraktowanie elementów zbioru uczącego pozwala podczas konstruowania klasyfikatora wykorzystać algorytmy podziału przestrzeni E^n na części, w obrębie których występują tylko punkty reprezentujące jedną z dwóch klas. Wśród takich algorytmów na uwagę zasługują algorytmy dzielące przestrzeń w sposób liniowy (tzn. przez hiperpłaszczyznę).

[1] William Shakespeare, *Hamlet, książe Danii*, tłum. Stanisław Barańczak, Wydawnictwo W drodze, Poznań 1990.

[2] Erich Fromm, *Mieć czy być*?, Dom Wydawniczy Rebis, Poznań 1999.

[3] klasyfikator ten może być traktowany jako klasyfikator odcinkowo-liniowy.

Takie podejście ma jedną ogromną zaletę — prosty model opisujący poszukiwaną hiperpłaszczyznę rozdzielającą; ma jednak również jedną zasadniczą wadę, która wyklucza użycie tego typu algorytmów w praktycznych zastosowaniach — rzeczywiste dane, w przeważającej większości, nie są liniowo rozdzielne w przestrzeni oryginalnej. Nie oznacza to jednak, że algorytm tego typu nie może być wykorzystany w konstrukcji hierarchicznego klasyfikatora liniowego. Powinien jednak spełniać pewne dodatkowe warunki.

Tezy pracy sformułujemy w następujący sposób:

> *Istnieje algorytm rozdzielania dwóch, skończonych zbiorów w sposób optymalny, jeśli jest możliwe rozdzielenie ich liniowo, o następujących własnościach: nie wymagana jest informacja a priori, czy zbiory są rozdzielne liniowo i przy każdym wykonaniu zwracany jest zawsze ten sam wynik dla określonych zbiorów.*
>
> *Własności te są wystarczające dla efektywnego wykorzystania wspomnianego algorytmu do konstrukcji hierarchicznego klasyfikatora odcinkowo-liniowego dla zbiorów, które nie są rozdzielne liniowo.*

W rozdziale pierwszym rozprawy przedstawiono różne, istniejące podejścia do tworzenia klasyfikatorów dwudecyzyjnych, ze szczególnym uwzględnieniem własności wykorzystywanych w nich algorytmów rozdzielających.

W rozdziale drugim przedstawiono zalety i wady istniejącego, opisanego w literaturze algorytmu, który stał się inspiracją zaproponowanego w rozdziale trzecim algorytmu zapewniającego optymalne rozdzielenie dwóch zbiorów. Przedstawiona konstrukcja algorytmu zachowała wszystkie zalety pierwowzoru, usuwając jego wady. Poprawność algorytmu została udowodniona w dodatku A. Pewne elementy przedstawionego dowodu pozwoliły na zoptymalizowanie numeryczne algorytmu, co zostało przedstawione w rozdziale czwartym.

W kolejnym, piątym rozdziale przedstawiono dwie idee konstruowania klasyfikatora odcinkowo-liniowego przeprowadzając dyskusję dotyczącą możliwości realizacji.

W ostatnim szóstym przedstawiono algorytm konstruowania hierarchicznego klasyfikatora liniowego, wykorzystującego algorytm przedstawiony w rozdziale trzecim.

W podsumowaniu przeprowadzono dyskusję dotyczącą możliwości wykorzystania przedstawionych algorytmów w praktycznych zastosowaniach.

Całość rozprawy stanowi jednocześnie swoisty „pamiętnik" stawianych sobie przez autorkę pytań i prób odpowiedzi na nie. W trakcie pisania rozprawy duży nacisk położono na uwzględnienie chronologii pojawiania się tych pytań. Ponieważ:

> „*Odpowiedź jest zawsze częścią drogi, którą masz już za sobą. Tylko pytanie może wskazać drogę naprzód* [...][4]"

uznano za interesujące opisanie również „ślepych" uliczek, którymi przyszło podążać autorce zanim trafiła na „właściwe" drogi. Wszystkie rozdziały kończą się pytaniem. Wszystkie rozdziały są odpowiedzią na pytania, które pojawiły się w poprzedzających je częściach, ale są jednocześnie źródłem nowych pytań.

[4] Jostein Gaarder, *Hej! Czy jest tu kto?*, tłum. Iwona Zimnicka, Jacek Santorski & Co i Wydawnictwo Wilga, 1997.

Spis treści

Rozdział 1

Metody wykorzystywane w zagadnieniach klasyfikacji

1.1. Czym są klasyfikatory?

W zagadnieniach poruszanych w ramach sztucznej inteligencji klasyfikatory mogą stanowić z jednej strony trzon poruszanego problemu (np. ze względu na swoje własności), ale mogą być również „elementem-narzędziem" o określonych i akceptowanych cechach, wykorzystywanym do rozwiązania szerzej postawionego problemu. I choć w pracy główny nacisk jest położony na konstrukcję klasyfikatora o określonych własnościach, nie zapomina się, że ma być on wykorzystany w praktycznych zastosowaniach.

Bez względu jednak na to jak traktujemy klasyfikatory, najbardziej pierwotnym pytaniem jest pytanie o zakres i znaczenie pojęcia *klasyfikator*. Otóż według słownika wyrazów obcych [1] klasyfikator to:

> „**1**. osoba zajmująca się klasyfikacją, dokonująca klasyfikacji **2**. urządzenie do rozdzielania materiału jednorodnego, złożonego z ziaren tej samej substancji (np. węgla), lecz o różnych rozmiarach na frakcje ziaren o tej samej, określonej wielkości."

Czym w takim razie jest według tego samego źródła *klasyfikacja*? To między innymi:

> „systematyczny podział przedmiotów lub zjawisk na klasy, działy, poddziały itd. według określonej zasady"

I choć opis ten jest daleki od terminów używanych w naukach ścisłych, to zawiera informacje o podstawowych cechach klasyfikatorów. Jednocześnie, niejako przy okazji, przytoczone definicje potwierdzają zasadność prowadzenia badań nad klasyfikatorami w ramach sztucznej inteligencji, jeżeli słowo „osoba" zamienimy np. na słowa, „program" czy „system informatyczny". Te istotne cechy to:

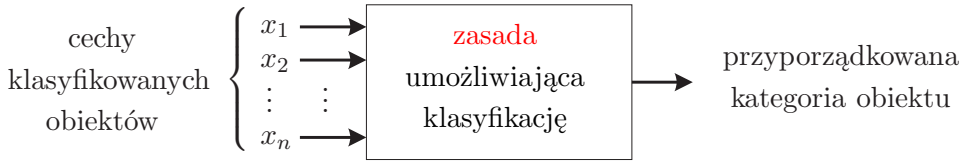
— **znajomość pewnych, wybranych cech** klasyfikowanych obiektów, na podstawie których dokonywana jest klasyfikacja;

— **znajomość klas**, do których możliwe jest przydzielenie klasyfikowanego obiektu (tzn. przedmiotu, zjawiska)

i co wydaje się najistotniejsze:

— **zasada**, według której obiekty są przypisywane do określonej kategorii.

Te trzy najważniejsze cechy klasyfikatora z uwzględnieniem wzajemnych zależności miedzy nimi przedstawiono w postaci ogólnego schematu na rysunku 1.1., zakładając bez utraty ogólności, że każdy z klasyfikowanych obiektów opisany jest skończoną liczbą n cech.



Rysunek 1.1: Ogólny schemat klasyfikatora

Przyjrzyjmy się najważniejszym elementom składowym klasyfikatorów z bliższej perspektywy.

W ogólności, każda z n cech reprezentujących klasyfikowany obiekt może być wielkością zarówno nominalną, jak i porządkową. Przy czym wśród tych ostatnich wyróżnia się podział na wartości dyskretne i ciągłe. Cechy w zależności od projektowanego systemu mogą być reprezentowane również jako pojęcia lingwistyczne.

Najbardziej zgrubna systematyka klasyfikatorów dzieli je na wielo-decyzyjne i dwu-decyzyjne (dychotomizatory) w zależności od tego, do ilu różnych klas potencjalnie może być zaklasyfikowany każdy z obiektów. Należy zaznaczyć, że klasyfikatory dwu-decyzyjne (inaczej binarne), z jednej strony są szczególnym przypadkiem klasyfikatorów wielo-decyzyjnych, z drugiej zaś, mogą być głównym „budulcem" umożliwiającym definiowanie wspomnianych klasyfikatorów wielo-decyzyjnych. W literaturze opisywane są trzy typowe podejścia [18]. I tak, klasyfikator umożliwiający klasyfikację obiektów należących do c klas można zastąpić m klasyfikatorami binarnymi, z których każdy umożliwia:

— stwierdzenie faktu, czy obiekt należy do i-tej klasy, czy też nie;
odpowiednio dla $i=1,2,\ldots,(c-1)$ — wówczas $m=c-1$;

— stwierdzenie faktu, czy obiekt należy do i-tej klasy, czy też nie;
odpowiednio dla $i=1,2,\ldots,c$ — wówczas $m=c$;

— klasyfikację do jednej z dwóch klas — i lub j,
gdzie $i\neq j$ dla $i,j=1,2,\ldots,c$ — wówczas $m=c\,(c-1)/2$.

I choć w tej samej pracy dwa ostatnie, z przedstawionych powyżej podejść, są krytykowane jako rozwiązania, które powodować mogą utworzenie dodatkowej kategorii obiektów „nierozpoznawalnych", to z praktycznego punktu widzenia, szczególnie w zastosowaniach medycznych, jest to kategoria jak najbardziej pożądana.

Jednak najważniejszym z omawianych elementów, który w największym stopniu decyduje o użyteczności klasyfikatora, jest ogólnie nazwana do tej pory *zasada* — mająca bezpośredni wpływ na to, do której klasy przydzielony zostanie klasyfikowany obiekt. Biorąc pod uwagę własności tego elementu klasyfikatory można podzielić na dwie grupy. Do pierwszej grupy należą klasyfikatory, gdzie wspomniana *zasada* jest znana, jednoznacznie określona i deterministyczna — wówczas raczej mówimy o urządzeniach [2]. Drugą grupę stanowią klasyfikatory, gdzie zasada nie jest znana *a priori* i właśnie ta grupa stanowi obszar zainteresowania sztucznej inteligencji.

Proces określania „zasady" nazywany jest *procesem uczenia klasyfikatora*. Zasygnalizujmy, że po przeprowadzeniu procesu uczenia utworzona reguła będzie w najlepszym razie określona i prawidłowa, ale nie oznacza to, że reguła ta będzie zrozumiała (w sensie interpretacji) dla człowieka.

Proces uczenia — inaczej „uczenie się” — oznacza, jak ładnie zostało to sformułowane w [7]:

„[. . .] dobranie na podstawie historycznych doświadczeń odpowiednich »parametrów«”

których wartości, w przypadku klasyfikatora, określać będą jednoznacznie „zasadę”. *Historyczne doświadczenia*, w tym kontekście, są zbiorem danych opisujących różne obiekty, których przynależność do jednej z reprezentowanych klas jest znana. Zbiór ten nazywany jest *zbiorem uczącym*. Opis pojedynczego obiektu (tzw. wzorca uczącego) jest przedstawiony w postaci opisanych wcześniej n cech.

W trakcie procesu uczenia klasyfikatora, informacja do jakiej klasy należą poszczególne obiekty ze zbioru uczącego, jest niezbędna z następujących powodów. Po pierwsze, do stwierdzenia czy na aktualnym etapie uczenia wzorzec uczący jest prawidłowo rozpoznany. Po drugie, informacja ta wykorzystywana jest w metodzie uaktualnienia »parametrów« klasyfikatora w jeden z następujących sposobów:

— **wprost** — wykorzystywana jest informacja o oczekiwanej przynależności obiektu — mamy do czynienia wtedy z tzw. uczeniem z nauczycielem;

— **pośrednio** — wykorzystywana jest tylko informacja o tym, czy obiekt został zaklasyfikowany poprawnie, czy błędnie — ten typ uczenia nazywany jest uczeniem ze wzmocnieniem.

W przypadku klasyfikatorów dwu-decyzyjnych, które są przedmiotem tej rozprawy, metody te można uznać za równoważne [18].

Bez względu na użytą metodę uaktualniania »parametrów« istotnym czynnikiem jest możliwość przeprowadzenia oceny jakości uczenia — czyli na ile otrzymana na pewnym etapie uczenia *zasada* spełnia nasze oczekiwania.

Najprostszym miernikiem jakości procesu uczenia określającego postępy „uczenia się” klasyfikatora, może być stosunek liczby błędnie klasyfikowanych wzorców ze zbioru uczącego do ogólnej liczby wzorców w zbiorze uczącym. Wartość ta nazywana jest ogólnym błędem klasyfikacji, a proces uczenia ma doprowadzić do minimalizacji tej wartości.

Więcej informacji dotyczących jakości uczenia dostarcza jednak macierz $\boldsymbol{E}=[e_{ij}]$, gdzie e_{ij} jest stosunkiem liczby obiektów z i-klasy rozpoznanych przez klasyfikator jako obiekty z j-klasy do liczby wszystkich obiektów ze zbioru uczącego należące do i-klasy, gdzie $i, j = 1, \ldots, c$ [28] (c — liczba klas). W tym przypadku oczekuje się, że proces uczenia doprowadzi do maksymalizacji wartości elementów znajdujących się na przekątnej macierzy $\boldsymbol{E}$ i minimalizacji elementów znajdujących się poza diagonalną macierzy.

Bez względu jednak na dobór miernika jakości procesu uczenia, ostatecznie interesuje nas „dobry” klasyfikator, a nie klasyfikator, który „dobrze się uczy”. Oznacza to, że z procesem uczenia i jego oceną nierozerwalnie związana jest również potrzeba weryfikacji otrzymanego klasyfikatora, którego *reguła klasyfikująca* jednoznacznie określona jest przez wyznaczone »parametry«. Naturalnym wydaje się dokonanie oceny klasyfikatora na danych, które nie zostały wykorzystane w procesie uczenia. Uwzględniając koszt i możliwość pozyskania dodatkowych danych, dostępne dane wykorzystane mogą być zarówno w procesie uczenia jak i ewaluacji klasyfikatora w jeden z następujących sposobów.

1. Dane dzielone są na dwa zbiory: uczący i testujący, przy czym z reguły większość elementów zbioru testującego nie znajduje się w zbiorze uczącym. W procesie uczenia wykorzystywany jest tylko zbiór uczący. Do ewaluacji klasyfikatora wykorzystywany jest tylko zbiór testujący.

2. CV, czyli n-krotna walidacja (*n-fold cross-validation*): dane dzielone są na n rozłącznych, w miarę równolicznych zbiorów (ich liczność różni się maksymalnie o jeden). Proces uczenia zostaje przeprowadzony $(n-1)$ razy, gdzie każdy z utworzonych zbiorów pełni rolę zbioru testującego, zaś pozostałe łącznie stanowią zbiór uczący.
3. 1-LEAVE-OUT — jest to szczególny przypadek metody CV, gdzie każdy z tworzonych n zbiorów jest jednoelementowy.

Każdy z wymienionych do tej pory czynników jest istotny z punktu widzenia przydatności projektowanego klasyfikatora, poczynając od przyjętej reprezentacji cech opisującej klasyfikowane obiekty, po konstrukcję zbiorów uczących i testujących, kończąc na poprawnej metodzie oceny procesu uczenia, czy klasyfikatora jako takiego. Każdy z tych czynników stanowić może przedmiot intensywnie prowadzonych badań [21, 22, 38, 39]. Żaden z nich nie jest jednak przedmiotem tej rozprawy.

Pozostał ostatni element, nazywany do tej pory, krótko »parametrami«. Co kryje się pod tym terminem?

1.2. »Parametry« klasyfikatora

Enigmatycznie oznaczane do tej pory »parametry« stanowią istotę uczenia klasyfikatorów i powinny w jednoznaczny sposób określają konkretny egzemplarz klasyfikatora.

Ze względu na szeroki wachlarz dostępnych metod nie sposób wymienić wszystkich możliwych znaczeń terminu »parametry«. Jako ilustrację poniżej przedstawiono wybrane przykłady.

W przypadku, gdy konstrukcja klasyfikatora wykorzystuje sztuczne sieci neuronowe, w ogólnym przypadku »parametry« pozwalają określić strukturę sieci (liczba warstw, sposób ich połączenia) oraz wartości wag neuronów znajdujących się w kolejnych warstwach [5]. Choć parametry te mają określone znaczenie, to w ogólności nie jest możliwa interpretacja, jakie zadanie w procesie klasyfikacji realizują poszczególne elementy składowe — neurony sieci.

Jeśli jednak do konstrukcji klasyfikatora użyjemy drzew decyzyjnych, które podobnie jak sieci neuronowe można traktować jako pewną strukturę złożoną z identycznych elementów, wówczas »parametry« bedą opisywać nie tylko strukturę drzewa. Dodatkowo każdy element składowy tej struktury będzie miał swoją interpretację. Parametry opisują bowiem hierarchię węzłów i liści drzewa, które zapewnia najlepszą klasyfikację wzorców znajdujących się w zbiorach uczącym i testującym. Każdy z węzłów przechowuje informację dotyczącą testu sprawdzającego wartość atrybutu (tj. cechy) dla cech nominalnych i porządkowych dyskretnych lub testu sprawdzającego czy wartość jest z określonego przedziału dla cech porządkowych (głównie ciągłych, rzadziej porządkowych dyskretnych). Każdy liść zawiera informację o przypisanej klasie. Testy znajdujące się w węzłach są testami prostymi, czyli nie zawierają koniunkcji czy alternatywy warunków dotyczących różnych cech.

»Parametry« mogą jednak opisywać nie tylko strukturę, której każdy z elementów składowych ma konkretną interpretację, ale również każdy z elementów może zawierać zarówno testy proste jak i testy złożone. Dzieje się tak w przypadku, gdy klasyfikator konstruowany jest przy użyciu zbioru reguł [7]. Wówczas »parametry« określają zapisane przy pomocy kompleksów reguły.

Jeśli zaś do opisu klasyfikatora użyjemy metod wykorzystujących oszacowania prawdopodobieństw wystąpienia w danych trenujących określonych wartości cech i klas to, w przeciwieństwie

do przytoczonych wcześniej przykładów, w skład »parametrów« będzie wchodził również sam zbiór uczący. Dzieje się tak w przypadku klasyfikatorów wykorzystujących twierdzenie Bayesa [7, 18]. W skrajnym przypadku np. *naiwnego* klasyfikatora bayesowskiego »parametry« ograniczają się tylko i wyłącznie do zbioru uczącego. Istnieje jeszcze jeden powód, dla którego o tej grupie metod należało wspomnieć. Otóż *optymalny* klasyfikator bayesowski (choć w praktyce nie stosowany) jest, jako najlepszy teoretycznie klasyfikator, punktem odniesienia do porównywania właściwości różnych metod konstrukcji klasyfikatorów.

W praktycznych zastosowaniach z tej grupy metod do konstrukcji klasyfikatora wykorzystywany jest algorytm k najbliższych sąsiadów, który w teoretycznych rozważaniach dla k dążącego do nieskończoności ma własności optymalnego klasyfikatora bayesowskiego. W tym przypadku »parametry« oznaczają zbiór uczący i wyznaczoną w procesie uczenia wartość k, która zapewnia najlepszą klasyfikację zbioru uczącego. Główne wady tego podejścia to potrzeba przechowywania zbioru uczącego i złożoność związana z klasyfikacją nowego obiektu. Jeżeli chodzi o próby usunięcia pierwszej z nich, znanych jest w literaturze wiele metod tworzenia tzw. *zbiorów odniesienia*, w skład których wchodzą tylko istotne z punktu widzenia klasyfikacji elementy ze zbioru uczącego [16, 35]. Stosowanie zbiorów odniesienia zmniejsza wagę drugiej wady, ale jej nie eliminuje.

Propozycja, która umożliwia usunięcie drugiej wady, a która stała się zalążkiem tej rozprawy jest zastosowanie hierarchicznego klasyfikatora odcinkowo-liniowego, który rozdziela liniowo elementy zbioru odniesienia. W takim przypadku, zachowamy własności klasyfikatora wykorzystującego algorytm k najbliższych sąsiadów, ale »parametry« będą opisywać hierarchię hiperpłaszczyzn rozdzielających, a proces klasyfikacji nowych obiektów będzie pozbawiony wymienionych wcześniej wad.

Pozostaje zdecydować się na algorytm liniowego rozdzielania dwóch zbiorów. Który z opisanych w literaturze algorytmów mógłby spełniać wymagania, które pozwolą zdefiniować hierarchiczny klasyfikator odcinkowo-liniowy? Jakie są te wymagania?

1.3. Metody badania liniowej rozdzielności dwóch zbiorów

1.3.1. Definicje „liniowej rozdzielności dwóch zbiorów"

Podstawowym elementem w proponowanym sposobie konstrukcji klasyfikatora odcinkowo-liniowego jest algorytm badania liniowej rozdzielności dwóch zbiorów. Jeśli n cech opisujących każdy z klasyfikowanych obiektów potraktujemy jako punkt w przestrzeni E^n a na zbiór uczący spojrzymy jako na sumę dwóch rozłącznych zbiorów X_1 i X_2, gdzie punkty ze zbioru X_1 należą do klasy pierwszej, zaś punkty ze zbioru X_2 do drugiej klasy. Wówczas przedstawione poniżej dwa stwierdzenia są równoważne i w dalszej części pracy mogą być używane wymiennie. Stwierdzenie pierwsze, które wynika bezpośrednio z natury postawionego problemu: „*poprawna klasyfikacja elementów zbioru uczącego przez projektowany klasyfikator*" jest równoważne, po przyjęciu proponowanego podejścia, stwierdzeniu, że „*zbiory X_1 i X_2 są rozdzielne liniowo*".

Przy wyborze odpowiedniego algorytmu pod uwagę należy wziąć nie tylko jego własności (np. złożoność obliczeniową, czy pamięciową), ale również wymagania dotyczące zbioru uczącego, których spełnienie jest niezbędne do poprawnego działania algorytmu. Zwróca się również uwagę na jednoznaczność i cechy otrzymywanych rozwiązań dla różnych wykonań algorytmu dla tego

samego zbioru uczącego. Określone, istotne własności otrzymywanego rozwiązania są powodem, dla którego na potrzeby rozprawy zdefiniujemy różne *typy* rozdzielności.

Definicja 1 *Zbiory X_1 i X_2 są rozdzielne liniowo, jeśli istnieje funkcja $g(\boldsymbol{x})$, taką że:*

$$\begin{cases} g(\boldsymbol{x}) \geqslant 0 & \text{gdy } \boldsymbol{x} \in X_1 \\ g(\boldsymbol{x}) \leqslant 0 & \text{gdy } \boldsymbol{x} \in X_2 \end{cases} \quad g(\boldsymbol{x}) = \sum_{i=1}^{n} a_i x_i + a, \quad \sum_{i=1}^{n} a_i^2 \neq 0 \tag{1.1}$$

Podkreślmy, że użycie funkcji $g(\boldsymbol{x})$ jako »parametrów« klasyfikatora nie musi zapewniać poprawnej klasyfikacji wszystkich elementów zbioru uczącego. Stanie się tak, jeśli na hiperpłaszczyźnie rozdzielającej $g(\boldsymbol{x})=0$ w przestrzeni E^n znajdą się co najmniej dwa punkty należące do różnych zbiorów X_1 i X_2.

Chcąc uniknąć tego typu sytuacji interesować nas będzie ścisła rozdzielność liniowa dwóch zbiorów. Z tym typem rozdzielności mamy do czynienia, jeśli poszukiwana funkcja odpowiada poniższej definicji.

Definicja 2 *Zbiory X_1 i X_2 są ściśle rozdzielne liniowo, gdy istnieje funkcja $g^*(\boldsymbol{x})$, taka że:*

$$\begin{cases} g^*(\boldsymbol{x}) > 0 & \text{gdy } \boldsymbol{x} \in X_1 \\ g^*(\boldsymbol{x}) < 0 & \text{gdy } \boldsymbol{x} \in X_2 \end{cases} \quad g^*(\boldsymbol{x}) = \sum_{i=1}^{n} a_i x_i + a, \quad \sum_{i=1}^{n} a_i^2 \neq 0 \tag{1.2}$$

przy czym relacja równości $g^*(\boldsymbol{x}) = 0$ może zachodzić tylko dla punktów należących do jednego ze zbiorów X_1 lub X_2.

Jeśli zbiory są ściśle rozdzielne, to istnieje nieskończenie wiele różnych, poprawnych rozwiązań. Wśród tych rozwiązań, wyróżnić można te, które zapewniają rozdzielność z *prześwitem* ε.

Definicja 3 *Zbiory X_1 i X_2 są rozdzielne liniowo z prześwitem ε, jeśli istnieje funkcja $h(\boldsymbol{x})$, taka że:*

$$\begin{cases} h(\boldsymbol{x}) \geqslant b & \text{gdy } \boldsymbol{x} \in X_1 \\ h(\boldsymbol{x}) \leqslant -b & \text{gdy } \boldsymbol{x} \in X_2 \end{cases} \quad h(\boldsymbol{x}) = \sum_{i=1}^{n} a_i x_i + a, \quad \sum_{i=1}^{n} a_i^2 \neq 0, \quad \varepsilon = \frac{2b}{\sum_{i=1}^{n} a_i^2} > 0 \tag{1.3}$$

W definicjach 1, 2 i 3 parametry opisujące poszukiwane funkcje, choć w konkretnych przypadkach będą przyjmować różne wartości, symbolicznie mogą być zapisane jako jeden wektor:

$$\boldsymbol{a} = [a_1,\ a_2,\ \ldots a_n,\ a] \in \mathrm{E}^{n+1} \tag{1.4}$$

W rozprawie proponujemy, jako podejście równoważne do definicji 3, zdefiniowanie ścisłej rozdzielności z prześwitem ε jako pary funkcji h_1 i h_2 gdzie, hiperpłaszczyzny wyznaczone przez $h(x)=0$, $h_1(x)=0$ i $h_2(x)=0$ są wzajemnie do siebie równoległe (definicja 4).

Definicja 4 *Zbiory X_1 i X_2 są ściśle rozdzielne liniowo z prześwitem ε, gdy istnieją funkcje $h_1(\boldsymbol{x})$ i $h_2(\boldsymbol{x})$ takie, że:*

$$\begin{cases} h_1(\boldsymbol{x}_1) \geqslant 0 \ \wedge \ h_2(\boldsymbol{x}_1) > 0 \\ h_1(\boldsymbol{x}_2) < 0 \ \wedge \ h_2(\boldsymbol{x}_2) \leqslant 0 \end{cases} \boldsymbol{x}_1 \in X_1, \boldsymbol{x}_2 \in X_2$$
$$h_1(\boldsymbol{x}) = \textstyle\sum_{i=1}^{n} a_i^* x_i + a_{n+2}^* \qquad \sum_{i=1}^{n} (a_i^*)^2 = 1, \quad \varepsilon > 0 \tag{1.5}$$
$$h_2(\boldsymbol{x}) = \textstyle\sum_{i=1}^{n} a_i^* x_i + \varepsilon + a_{n+2}^*$$

Obszar wyznaczony przez nierówności: $h_1(\boldsymbol{x})<0$ i $h_2(\boldsymbol{x})>0$ dla każdego $\boldsymbol{x}\in E^n$ nazywać będziemy prześwitem (marginesem).

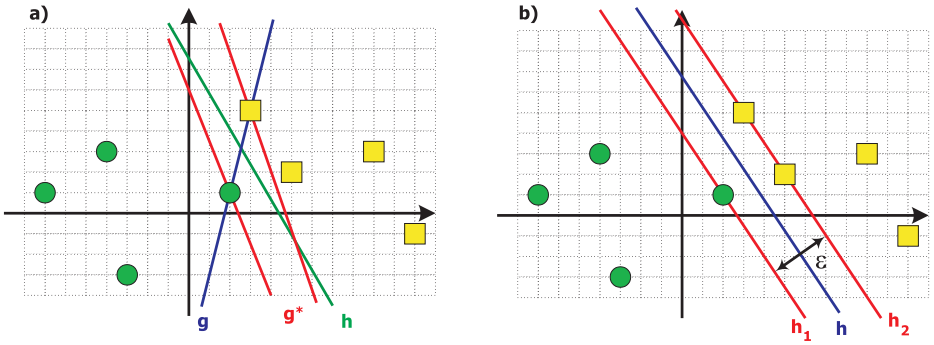
Wśród rozwiązań zapewniających rozdzielność z prześwitem najbardziej zainteresowani jesteśmy tymi rozwiązaniami, które zapewniają maksymalną wartość prześwitu. Wówczas będziemy mówić o optymalnej rozdzielności dwóch zbiorów.

Definicja 5 *Zbiory X_1 i X_2 są rozdzielne optymalnie z prześwitem ε jeśli istnieje para hiperpłaszczyzn rozdzielających, maksymalnie od siebie odległych, a więc istnieją funkcje $h_1(\boldsymbol{x})$ i $h_2(\boldsymbol{x})$ takie, że:*

$$\begin{cases} h_1(\boldsymbol{x}_1) \geqslant 0 \quad \wedge \quad h_2(\boldsymbol{x}_1) > 0 \\ h_1(\boldsymbol{x}_2) < 0 \quad \wedge \quad h_2(\boldsymbol{x}_2) \leqslant 0 \end{cases} \quad \boldsymbol{x}_1 \in X_1, \boldsymbol{x}_2 \in X_2 \tag{1.6}$$

$$\begin{array}{ll} h_1(\boldsymbol{x}) = \sum_{i=1}^{n} a_i^* x_i + a_{n+2}^* & \\ h_2(\boldsymbol{x}) = \sum_{i=1}^{n} a_i^* x_i + \varepsilon + a_{n+2}^* & \sum_{i=1}^{n} (a_i^*)^2 = 1, \quad max(\varepsilon) \end{array}$$

Na rysunku 1.3.1. przedstawiono przykłady zdefiniowanych powyżej typów rozdzielności liniowej dwóch zbiorów w przestrzeni E^2.



Rysunek 1.2: Przykłady różnych *typów* rozdzielności: a) definicja 1 (proste g, g^*, h), definicja 2 (proste g^*, h), definicja 3 (prosta h); b) definicja 3 (prosta h), definicja 4 (proste h_1, h_2).

Biorąc pod uwagę wymienione typy rozdzielności liniowej możemy przystąpić do przeglądu metod umożliwiających znalezienie parametrów określających hiperpłaszczyznę rozdzielającą.

1.3.2. Funkcja dyskryminująca Fishera (Fisher's discriminant function)

Wykorzystanie funkcji liniowych w konstrukcji klasyfikatorów historycznie związane jest z osobą Ronalda A. Fishera [33] Choć podejście, które zaproponował, częściej cytowane jest jako przykład wstępnej obróbki danych, gdzie celem jest redukcja liczby cech opisujących klasyfikowane obiekty, to jak pokazano w [18] może być zastosowane jako algorytm wyznaczenia hiperpłaszczyzny rozdzielającej zgodnie z definicją 1 (wzór(1)). Przy czym, otrzymanie poprawnego, w tym sensie, rozwiązania jest możliwie tylko w przypadku, gdy dane są rzeczywiście liniowo rozdzielne. Wiąże się to z ideą wyznaczania wektora normalnego hiperpłaszyzny rozdzielającej. Wyznaczany wektor $[a_1, \ldots, a_n]$ leży na takiej prostej w przestrzeni E^n, która zapewnia, że punkty ze zbiorów X_1' i X_2', będących zbiorami rzutów na tą prostą odpowiadających im punktów ze zbiorów X_1 i X_2, charakteryzują się maksymalnym zróżnicowaniem międzyklasowym tzn.:

$$\max\left(|m_1 - m_2|\right), \qquad \text{gdzie} \quad m_i = \frac{1}{\operatorname{card}\left(X_i'\right)} \sum_{x' \in X_i'} x' \quad \text{dla } i = 1, 2;$$

oraz minimalną wariancją wewnątrzklasową tj.:

$$v_i^2 = \sum_{x' \in X_i'} \left(x' - m_i\right)^2 \quad \text{dla } i = 1, 2.$$

Idea kryjąca się w tym podejściu wzbogacona o wykorzystanie analizy składowych głównych umożliwia konstrukcję klasyfikatorów wielodecyzyjnych [18] lub może być stosowana jako element obróbki wstępnej danych (*data preprocessing*) do wyboru najbardziej znaczących cech [15, 40, 43].

Niewątpliwą zaletą przedstawionej metody, z punktu widzenia praktycznych zastosowań, jest fakt, że dla określonego zbioru uczącego otrzymujemy zawsze dokładnie ten sam wynik.

Z drugiej jednak strony otrzymujemy tylko jedno rozwiązanie, które albo jest „odpowiednie", albo nie — nie znamy „kierunku" w jakim należałoby poszukiwać „lepszego" rozwiązania. Na przeciw takiemu ujęciu problemu wychodzi grupa metod, które w skrócie w literaturze nazywane są metodami gradientowymi.

1.3.3. Metody gradientowe

Metody tego typu, jako przykład metod iteracyjnych, do wyznaczenia poszukiwanego wektora $\boldsymbol{a}$, wykorzystują nie tylko zbiór uczący, ale charakteryzują się dodatkowymi parametrami, od których zależy jakość i szybkość otrzymanego rozwiązania. Parametry te pozwalają określić kierunek zmian aktualnego, otrzymanego w i-tym kroku, wektora $\boldsymbol{a}^{(i)}$, „tempo" tych zmian oraz warunek zatrzymania.

Kierunek zmian wyznaczany jest przez gradient funkcji (kryterium) błędu. Funkcja błędu $J(\boldsymbol{a}^{(i)})$ może być dowolną funkcją klasy C^1, ale wymagane jest, aby osiągała minimum, jeśli wektor $\boldsymbol{a}^{(i)}$ jest takim poszukiwanym, a jeszcze nieznanym rozwiązaniem $\boldsymbol{a}$, które umożliwia poprawną klasyfikację wszystkich elementów zawartych w zbiorze uczącym.

„Tempo" zmian określane jest przez współczynnik uczenia η, który może być wartością stałą, lub zmienną w czasie poszukiwania rozwiązania w zależności od wykorzystanej metody [18].

Poszukiwany wektor, opisujący hiperpłaszczyznę rozdzielającą, w kolejnych iteracjach wyznaczany jest według ogólnej reguły:

$$\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} - \eta(i)\ \nabla J\Big(\boldsymbol{a}^{(i)}\Big) \tag{1.7}$$

gdzie wartość początkowa wyznaczanego wektora $\boldsymbol{a}^{(0)}$ najczęściej jest zupełnie dowolna. Metoda kończy swoje działanie, gdy wielkość korekty $\left(\boldsymbol{a}^{(i+1)} - \boldsymbol{a}^{(i)}\right)$ lub samej funkcji J jest mniejsza od ustalonej wartości θ, która stanowi główne kryterium stopu.

Jeśli definicję rozdzielności określonej wzorem (1.1), opisanej dwoma nierównościami, przedstawimy w postaci jednego warunku, stosując przekształcenie:

$$f(\boldsymbol{x}) = \begin{cases} [x_1, x_2, \ldots x_n, 1] & \text{gdy } \boldsymbol{x} \in X_1 \\ [-x_1, -x_2, \cdots - x_n, -1] & \text{gdy } \boldsymbol{x} \in X_2 \end{cases}, \qquad \text{dla } \boldsymbol{x} \in X = X_1 \cup X_2 \tag{1.8}$$

to warunek ten będzie miał postać:

$$\langle \boldsymbol{a}, \boldsymbol{t} \rangle \geqslant 0, \quad \text{dla } \boldsymbol{t} \in T = \{\boldsymbol{x} \in X_1 \cup X_2 : f(\boldsymbol{x})\} \tag{1.9}$$

gdzie $\langle ., . \rangle$ oznacza iloczyn skalarny.

Zastosowanie przekształcenia (1.8) umożliwia również sprawne zapisanie funkcji błędu. Przytoczmy, jako prosty przykład funkcji błędu, funkcję wykorzystywaną w metodzie perceptronowej:

$$J\Big(\boldsymbol{a}^{(i)}\Big) = 0 \ - \sum_{\substack{\boldsymbol{t} \in T \\ \langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \rangle < 0}} \Big\langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \Big\rangle \tag{1.10}$$

Podkreślmy, że wartość tej funkcji błędu zależy tylko i wyłącznie od niepoprawnie sklasyfikowanych wzorców zbioru uczącego.

W tym przypadku, podczas kolejnej iteracji metody (w trybie *batch-mode*) poszukiwany wektor przyjmuje wartość:

$$\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} + \eta(k) \sum_{\substack{\boldsymbol{t} \in T \\ \langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \rangle < 0}} \boldsymbol{t} \tag{1.11}$$

W literaturze [18] opisano wiele modyfikacji przedstawionej metody, od metody w wersji *on-line*, poprzez metodę, w której współczynnik η jest stały i wynosi 1 (*fixed-increment rule*), po metodę, która poszukuje rozwiązania z pewną wartością prześwitu ε zgodnego z definicją 4 (wzór 1.3). Wszystkie one jednak stanowią podgrupę metod gradientowych nazywanych *error--correcting procedures*, ze względu na fakt, że wektor wag ulega zmianie tylko w przypadku, nieprawidłowego zaklasyfikowania wzorca $\boldsymbol{t}$. Metody tego typu minimalizują liczbę źle sklasyfikowanych elementów zbioru uczącego.

Zamieszczony przykład funkcji błędu (1.10) podano, aby pokazać, że wartość, o którą zostaje skorygowany wektor $\boldsymbol{a}^{(i)}$, jest zdominowana przez najdłuższy źle sklasyfikowany wektor $\boldsymbol{t}$. Przykładem funkcji błędu, która pozbawiona jest tej wady, jest funkcja używana w metodach opisanych w literaturze jako metoda relaksacyjna (*relaxation procedure*), zapisana wzorem (1.12), gdy przy pomocy metody gradientowej poszukiwane jest rozwiązanie mające zapewniać rozdzielność zgodną z definicją 2 oraz wzorem (1.13), kiedy poszukiwane rozwiązanie ma rozdzielać zbiory z prześwitem ε zgodnie z definicją 4.

$$J\Big(\boldsymbol{a}^{(i)}\Big) = 0 + \frac{1}{2} \sum_{\substack{\boldsymbol{t} \in T \\ \langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \rangle < 0}} \frac{\Big\langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \Big\rangle^2}{\langle \boldsymbol{t}, \boldsymbol{t} \rangle} \tag{1.12}$$

$$J_{\varepsilon}\Big(\boldsymbol{a}^{(i)}\Big) = 0 + \frac{1}{2} \sum_{\substack{\boldsymbol{t} \in T \\ \langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \rangle < \frac{\varepsilon}{2}}} \frac{\Big(\Big\langle \boldsymbol{a}^{(i)}, \boldsymbol{t} \Big\rangle - \frac{\varepsilon}{2}\Big)^2}{\langle \boldsymbol{t}, \boldsymbol{t} \rangle} \tag{1.13}$$

Dla wszystkich wymienionych metod istnieją dowody zbieżności do poprawnego rozwiązania [18]. W tym samym źródle wykazano również, że bez względu na wybór początkowej wartości wektora $\boldsymbol{a}^{(0)}$ w skończonej liczbie kroków uzyskamy rozwiązanie, pod jednym jednak warunkiem, że zbiory są faktycznie rozdzielne liniowo, czego w przypadku rzeczywistych danych najczęściej nie wiemy *a priori*.

Wspomniane ograniczenie wyklucza możliwość użycia tych metod do konstrukcji hierarchicznego klasyfikatora odcinkowo-liniowego, który z założenia o „hierarchiczności" zakłada, że dane nie będą rozdzielne liniowo.

Drugą podgrupę metod, które można zaliczyć do grupy metod gradientowych stanowią te, które zamiast minimalizować liczbę błędnych klasyfikacji, minimalizują błąd średni kwadratowy pomiędzy pożądaną, arbitralnie ustaloną dodatnią wartością (b_i) a wartością iloczynu skalarnego $\langle \boldsymbol{t}_j, \boldsymbol{a} \rangle$ dla każdego wzorca $\boldsymbol{t}_j \in T$, $j = 1, \ldots, p = \mathrm{card}(T)$. Poszukiwany wektor $\boldsymbol{a}$ ma więc spełniać

następujące równanie macierzowe:

$$\begin{bmatrix} t_{1,1} & \dots & t_{1,n} & t_{1,n+1} \\ t_{2,1} & \dots & t_{2,n} & t_{2,n+1} \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ t_{p,1} & \dots & t_{p,n} & t_{p,n+1} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ a \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_p \end{bmatrix} \tag{1.14}$$

które w skrócie możemy zapisać jako:

$$\boldsymbol{T}\boldsymbol{a} = \boldsymbol{b}. \tag{1.14}$$

Funkcja błędu w tym przypadku przyjmować może następującą postać:

$$J(\boldsymbol{a}) = \|\boldsymbol{T}\boldsymbol{a} - \boldsymbol{b}\|^2 \tag{1.15}$$

Określenie wartości poszukiwanego wektora $\boldsymbol{a}$, który minimalizuje funkcję błędu, może być wykonane przy pomocy rachunku macierzowego (przy pewnych dodatkowych założeniach) [18] lub poprzez zastosowanie metody gradientowej w postaci metody iteracyjnej. Ta druga nazywana jest w literaturze regułą Widrowa-Hoffa lub regułą LMS (*least-mean-squared*) i do złudzenia, pomijając czynnik skalujący, przypomina metodę relaksacyjną (1.13), jeśli funkcję błędu (1.15) przedstawimy w postaci iloczynu skalarnego:

$$J\Big(\boldsymbol{a}^{(i)}\Big) = \sum_{\substack{\boldsymbol{t}_j \in T \\ j=1,\dots,p}} \Big(\Big\langle \boldsymbol{a}^{(i)}, \boldsymbol{t}_j \Big\rangle - b_j\Big)^2 \tag{1.16}$$

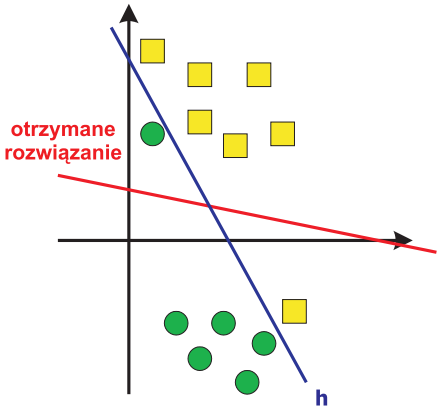
Choć podejście w tej metodzie zasadniczo różni się od idei wykorzystanej w metodzie perceptronowej, to z punktu widzenia projektowanego algorytmu tworzenia klasyfikatora hierarchicznego, tego typu metoda również nie może być algorytmem bazowym. Pomimo, że w odróżnieniu do wcześniej przedstawionych metod, w tym przypadku zawsze otrzymamy rozwiązanie prawidłowe, w sensie minimalnej wartości kwadratów odpowiednich różnic, to własności otrzymanego rozwiązania zależą od przyjętych wartości składowych wektora $\boldsymbol{b}$ i nie ma pewności, że otrzymane rozwiązanie pozwoli na poprawną klasyfikację elementów zbioru uczącego, nawet jeśli są to zbiory rozdzielne liniowo. Przykład ilustrujący sytuację tego typu przedstawiono na rysunku 1.3.

Niewątpliwą zaletą metody LMS, jest fakt, że zawsze otrzymamy „poprawne" rozwiązanie. Ogromną zaś wadą jest przymus określania wektora $\boldsymbol{b}$. Wady tej pozbawiona jest metoda będąca modyfikacją reguły Widrowa-Hoffa przedstawiona przez Ho'a i Kashyapa [24] i nazywana w literaturze metodą Ho-Kashyap.

Wprowadzone zmiany dotyczyły po pierwsze, własności wektora $\boldsymbol{b}$. Wszystkie jego składowe początkowe muszą mieć wartości dodatnie, ale w trakcie iteracji mogą ulec zmianie, co zwalnia nas z podjęcia decyzji, jakie ich wartości byłyby najbardziej korzystne z punktu widzenia rozwiązania, którego poszukujemy.

Po drugie, idea przyświecająca wprowadzonym zmianom była następująca: interesuje nas nie jeden wektor $\boldsymbol{a}$, który minimalizuje funkcję (1.15), ale dwa wektory $\boldsymbol{a}$ i $\boldsymbol{b}$, które minimalizują wspomnianą funkcję. Oznacza to, że metoda gradientowa powinna uwzględniać gradient wyznaczony ze względu na $\boldsymbol{a}$:

$$\nabla_{\boldsymbol{a}} J = 2\boldsymbol{T}^t(\boldsymbol{T}\boldsymbol{a} - \boldsymbol{b}) \tag{1.17}$$



Rysunek 1.3: Przykład użycia metody gradientowej z regułą Widrowa-Hoffa [18].

oraz gradient wyznaczony ze względu na $\boldsymbol{b}$:

$$\nabla_{\boldsymbol{b}} J = -2(\boldsymbol{T}\boldsymbol{a} - \boldsymbol{b}) \tag{1.18}$$

Jeżeli zaś dla dowolnego wektora $\boldsymbol{b}$ przyjmiemy wektor $\boldsymbol{a}$ równy:

$$\boldsymbol{a} = \boldsymbol{T}^{*}\boldsymbol{b} \tag{1.19}$$

gdzie $\boldsymbol{T}^{*} = (\boldsymbol{T}^{t}\boldsymbol{T})^{-1}\boldsymbol{T}^{t}$ jest pseudoodwrotną macierzą do macierzy $\boldsymbol{T}$, to zapewniamy tym samym w jednym kroku, że $\nabla_{\boldsymbol{a}} J = 0$.

W metodzie wprowadzone ograniczenia dotyczące wektora $\boldsymbol{b}$, dotyczą nie tylko inicjalnych wartości jego składowych. Podczas każdej iteracji, z założenia, wektor ten musi mieć wartości dodatnie. Ponieważ poszczególne składniki wektora $\nabla_{\boldsymbol{b}} J$ (wzór 1.18) w ogólności mogą być dowolnego znaku, aby zapewnić dodatniość składników wyznaczanego wektora $\boldsymbol{b}^{(i+1)}$ stosowany jest następujący wzór:

$$\boldsymbol{b}^{(i+1)} = \boldsymbol{b}^{(i)} - \eta(i)\,\tfrac{1}{2}\,[\,\nabla_{\boldsymbol{b}} J - |\nabla_{\boldsymbol{b}} J|\,] \tag{1.20}$$

Oznacza to, że w kolejnych iteracja poszczególne składniki wektora $\boldsymbol{b}^{(i+1)}$ nie zmienią się lub zwiększą swoją wartość.

Na pojedynczą iterację składają się więc następujące kroki:

1. Wyznaczenie wektora błędu $\boldsymbol{e}^{(i)} = \boldsymbol{T}\boldsymbol{a}^{(i)} - \boldsymbol{b}^{(i)}$.
2. Wyznaczenie wektora korekty $\boldsymbol{c}_b^{(i)} = \frac{1}{2}\left(\boldsymbol{e}^{(i)} + \left|\boldsymbol{e}^{(i)}\right|\right)$.
3. Wyznaczenie wektora $\boldsymbol{b}^{(i+1)} = \boldsymbol{b}^{(i)} + 2\,\eta(i)\,\boldsymbol{c}_b^{(i)}$.
4. Wyznaczenie wektora $\boldsymbol{a}^{(i+1)} = \boldsymbol{T}^{*}\boldsymbol{b}^{(i+1)}$.

W przypadku, gdy macierz $\boldsymbol{T}^{t}\boldsymbol{T}$ jest macierzą osobliwą, istnieje modyfikacja metody Ho-Kashyap, w której podane powyżej kroki 3–4 maja następującą postać:

3′. Wyznaczenie wektora $\boldsymbol{b}^{(i+1)} = \boldsymbol{b}^{(i)} + 2\,\boldsymbol{c}_b^{(i)}$.

4′. Wyznaczenie wektora $\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} + \eta\,\boldsymbol{D}\,\boldsymbol{T}^{t}\left|\boldsymbol{e}^{(i)}\right|$.

gdzie macierz $\boldsymbol{D}$ jest dowolną macierzą dodatnio określoną o wymiarach $(n+1) \times (n+1)$.

Jak wykazano w pracach [4, 18] zarówno oryginalna wersja metody Ho-Kashyap, jak i jej modyfikacje mają jedną poważna wadę z punktu widzenia praktycznego zastosowania w konstrukcji klasyfikatora hierarchicznego — nie jest znane żadne oszacowanie górne maksymalnej liczby kroków, aby stwierdzić fakt, że zbiory nie są rozdzielne liniowo.

1.3.4. Programowanie liniowe — metoda sympleksów

Zupełnie innym podejściem, jest zastosowanie metod programowania liniowego. Jest to podejście zdecydowanie bardziej złożone od przedstawionych metod gradientowych, ale zapewnia znalezienie prawidłowowego rozwiązania lub stwierdzenia faktu, że zbiory nie są rozdzielne liniowo w skończonej liczbie kroków.

Pozostaje tylko problem poszukiwania hiperłaszczyzny rozdzielającej sprowadzić do formuły wykorzystywanej w programowaniu liniowym. Należy więc zdefiniować funkcję celu $f_c(\tilde{\boldsymbol{a}})$, której minimalizacją jesteśmy zainteresowani, jako funkcję wyznaczanego wektora $\tilde{\boldsymbol{a}}$, oraz przedstawić ograniczenia w postaci nierówności.

Należy jednak pamiętać, że w programowaniu liniowym narzucone jest dodatkowe ograniczenie dotyczące poszukiwanego wektora $\tilde{\boldsymbol{a}}$ — mianowicie wszystkie jego składniki powinny być nieujemne. Z tego powodu, należy problem wyznaczania interesującego nas wektora $\boldsymbol{a}$, który w ogólności może mieć składowe o dowolnym znaku, sprowadzić do wyznaczania wektora $\tilde{\boldsymbol{a}}$, który spełnia to dodatkowe ograniczenie a jednocześnie umożliwia wyznaczenie wektora $\boldsymbol{a}$.

Definiując wektor określający hiperpłaszyczną rozdzielającą jako różnicę dwóch wektorów o czynnikach nieujemnych tzn.:

$$\boldsymbol{a} = \boldsymbol{a}^+ - \boldsymbol{a}^- \tag{1.21}$$

gdzie:

$$\boldsymbol{a}^+ = \tfrac{1}{2}\left(|\boldsymbol{a}| + \boldsymbol{a}\right) \tag{1.22}$$

$$\boldsymbol{a}^- = \tfrac{1}{2}\left(|\boldsymbol{a}| - \boldsymbol{a}\right) \tag{1.23}$$

i zakładając, że wektory $\boldsymbol{a}^+$ i $\boldsymbol{a}^-$ są częścią poszukiwanego przez metodę sympleksów wektora $\tilde{\boldsymbol{a}}$, czynimy zadość temu dodatkowemu założeniu, nie tracąc możliwości otrzymania w rezultacie wektora $\boldsymbol{a}$ o dowolnych wartości jego składowych.

Nierówności określające ograniczenia uzyskamy w następujących dwóch krokach. Po pierwsze, przedstawimy równość (1.14) w postaci nierówności (1.24), przy czym wprowadzimy dodatkowy, *sztuczny* parametr $\boldsymbol{\tau} = [\tau, \tau, \ldots, \tau]$, którego istnienie zostanie wykorzystane w definicji funkcji celu.

$$\boldsymbol{T}\boldsymbol{a} + \boldsymbol{\tau} \geqslant \boldsymbol{b} \tag{1.24}$$

Po drugie, uwzględnimy fakt, że wyznaczanym wektorem w metodzie sympleksów będzie wektor $\tilde{\boldsymbol{a}}$, a nie $\boldsymbol{a}$:

$$\tilde{\boldsymbol{T}}\,\tilde{\boldsymbol{a}} \geqslant \boldsymbol{b} \tag{1.25}$$

gdzie:

$$\tilde{\boldsymbol{T}} = \left[\begin{array}{|c|}\hline \\ \boldsymbol{T} \\ \\ \hline\end{array}\begin{array}{|c|}\hline \\ -\boldsymbol{T} \\ \\ \hline\end{array}\begin{array}{c} 1 \\ 1 \\ \vdots \\ \vdots \\ \vdots \\ 1 \end{array}\right], \qquad \tilde{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{a}^+ \\ \boldsymbol{a}^- \\ \tau \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_p \end{bmatrix}$$

Definiując dodatkowo wektor $\boldsymbol{\alpha}^t = [\mathbf{0},\, \mathbf{0},\, 1]$ funkcję celu możemy zdefiniować jako:

$$f_c(\tilde{\boldsymbol{a}}) = \boldsymbol{\alpha}^t\, \tilde{\boldsymbol{a}} \tag{1.26}$$

Składowe wektora $\tilde{\boldsymbol{a}}$, po wykonaniu metody sympleksów, pozwalają wyznaczyć poszukiwany wektor $\boldsymbol{a}$ zgodnie ze wzorem (1.21). Wartość ostatniej składowej τ pozwala stwierdzić, czy zbiory są, czy nie są rozdzielne liniowo. Wartość ta jest dodatnia, gdy zbiory nie są rozdzielne, w przeciwnym przypadku wynosi zero.

Powodem tego, że metoda ta nie cieszy się powodzeniem są wymagania pamięciowe. W przypadku obiektów opisanych przez n cech, liczba poszukiwanych parametrów wynosi $2n{+}3$ a liczba ograniczeń jest równa liczbie wzorców w zbiorze uczącym.

1.3.5. Techniki SVM

Historycznie, techniki SVM[1] — techniki „wektorów wspierających" (lub podpierających) — z przedstawianych w tym rozdziale metod są najmłodsze [9, 41, 42].

W technikach tych wykorzystywana jest definicja 3 rozdzielności dwóch zbiorów z przyjętą na stałe wartością $b{=}1$ (wzór 1.3). Jeżeli, podobnie jak w przypadku metod gradientowych, dokonamy przekształcenia (1.8), wówczas warunki wspomnianej definicji można zapisać w postaci jednego warunku:

$$\begin{gathered} \langle \boldsymbol{a}, \boldsymbol{t} \rangle \geqslant 1, \quad \text{dla } \boldsymbol{t} \in T = \{\boldsymbol{x} \in X_1 \cup X_2 : f(\boldsymbol{x})\} \\ \boldsymbol{a} = [\boldsymbol{a}',\, a] = [a_1,\, a_2,\, \ldots a_n,\, a] \in \mathrm{E}^{n+1} \end{gathered} \tag{1.27}$$

Elementy $\boldsymbol{t}$, dla których w powyższym warunku zachodzi równość, nazywane są *wektorami wspierającymi*.

Metody wektorów wspierających umożliwiają wyznaczenie wektora $\boldsymbol{a}$ zapewniającego ścisłą, optymalną, czyli o maksymalnej wartości prześwitu, rozdzielność dwóch zbiorów, jeśli jest to możliwe. Biorąc pod uwagę wzór (1.3), oznacza to, że poszukiwany wektor ma spełniać warunek (1.27) i jednocześnie minimalizować wyrażenie:

$$\sum_{i=0}^{n} a_i^2 = \langle \boldsymbol{a}',\, \boldsymbol{a}' \rangle \tag{1.28}$$

co zapewnia maksymalną wartość prześwitu, równą:

$$\varepsilon = \frac{2b}{\langle \boldsymbol{a}',\, \boldsymbol{a}' \rangle} \tag{1.29}$$

[1] *Support Vector Machines.*

Możemy więc na problem wyznaczenia wektora $\boldsymbol{a}' \in E^n$ i skalara a, które stanowią poszukiwany wektor $\boldsymbol{a} \in E^{n+1}$ (1.27), spojrzeć jako na zagadnienie programowania kwadratowego, gdzie należy zminimalizować funkcję:

$$f_c(\boldsymbol{a}') = \langle \boldsymbol{a}', \boldsymbol{a}' \rangle \tag{1.30}$$

z ograniczeniami w postaci nierówności:

$$\langle \boldsymbol{a}, \boldsymbol{t} \rangle \geqslant 1, \qquad \boldsymbol{a} = [\boldsymbol{a}', a] \tag{1.31}$$

Zagadnienie tego typu rozwiązuje się zazwyczaj poprzez wprowadzenie nieujemnych parametrów α_p (p=card(T)), zwanych mnożnikami Lagrange'a i funkcji Lagrange'a:

$$L(\boldsymbol{a}, \boldsymbol{\alpha}) = \tfrac{1}{2} \langle \boldsymbol{a}', \boldsymbol{a}' \rangle - \sum_{i=1}^{p} \alpha_i \left(\langle \boldsymbol{a}, \boldsymbol{t}_i \rangle - 1 \right) \tag{1.32}$$

przy dodatkowym warunku dotyczącym parametrów $\alpha_i \geqslant 0$ dla $i = 1, 2, \ldots, p$. Celem jest teraz maksymalizacja lagrangianu ze względu na współczynniki α_i i minimalizacja ze względu na $\boldsymbol{a}$ [9, 21, 41]. Istotne jest to, że tak zdefiniowany problem może nie mieć rozwiązania, dla zbiorów nierozdzielnych liniowo.

Korzystając jednak z twierdzenia Karush-Kuhn-Thuckera [9, 21], możemy przekształcić ten problem na dualny problem optymalizacyjny, który dotyczy wyłącznie maksymalizacji czynników α_i, gdzie czynnikami są iloczyny skalarne $\langle t_i, t_j \rangle$ dla $i, j = 1, \ldots, p$. Możliwość ta jest najcenniejszą własnością wartości technik SVM, pozwala bowiem na wprowadzenie *funkcji bazowych* (*jąder*)[2] [21] a następnie funkcji $K_k(\boldsymbol{t_i}, \boldsymbol{t_j}) = \langle \phi_k(\boldsymbol{t}_i), \phi_k(\boldsymbol{t}_j) \rangle$, które zastępują wspomniane iloczyny skalarne. Zastosowanie wielu funkcji bazowych $\phi(\boldsymbol{t})$, gdzie $\phi : E^n \to P^*$ jest nieliniową transformacją, powoduje, że szukamy rozwiązania w przestrzeni P^*, której wymiar jest dużo większy niż oryginalny wymiar wzorców w zbiorze uczącym. Modyfikacja ta powoduje, że w przypadkach, gdy zbiory nie są rozdzielne liniowo w oryginalej przestrzeni, dzięki zastosowaniu funkcji bazowych są rozdzielne w przestrzeni P^*.

Pojawia się jednak pytanie, czy jest możliwe poszukiwanie rozwiązania w przestrzeni zbliżonej do oryginalnej, a nie o wymiarowości dużo większej? Pytanie to ma uzasadnienie w związku z oszacowaniem [42] średniej wartości prawdopodobieństwa błędnej klasyfikacji P(error) używając do ewaluacji metody 1-LEAVE-OUT w zależności od liczby wektorów wspierających:

$$\mathrm{E}\left[P(\text{error})\right] = \frac{\mathrm{E}\left[\text{liczba wektorów wspierających}\right]}{\text{card}(T) - 1} \tag{1.33}$$

Im większa wymiarowość przestrzeni P^*, tym więcej punktów może być wektorami wspierającymi. Z tego powodu jesteśmy zainteresowani zmniejszeniem wymiarowości przestrzeni P^*.

Nie ma jednak gwarancji, że zbiory muszą być rozdzielne liniowo nawet w przestrzeni P^*. W pracy [8] przedstawiono inną definicję problemu, którego rozwiązanie w praktyce oznacza to, że dopuszcza się, aby niektóre z wzorców uczących były źle klasyfikowane.

1.3.6. Inne metody

Przedstawione do tej pory metody optymalizują zawsze jakąś funkcję. Inne podejście zostało zaproponowane przez Kozinca w pracy [30]. Traktuje on wzorce uczące każdej z klas jako punkty w przestrzeni E^n, które wyznaczają dwie powłoki wypukłe $\tilde{X}_1$ i $\tilde{X}_2$. Poszukiwany wektor $\boldsymbol{a}$

[2] *kernel functions.*

obliczany jest na podstawie wyznaczonych dwóch najbliżej położonych względem siebie punktów, z których pierwszy należy do powłoki $\tilde{X}_1$ zaś drugi do $\tilde{X}_2$. Ponieważ, jak pokazano w pracy [20], algorytm ten w szczególnym przypadku staje się metodą perceptronową, oznacza to, że algorytm ten jest przydatny tylko dla zbiorów rozdzielnych liniowo.

To geometryczne spojrzenie na problem było inspiracją do podejścia, które choć historycznie starsze od technik SVM, łączy ideę Kozinca z wyborem wektorów wspierajacych. Metodę tę przedstawił Jóźwik w pracy [25, 26]. Ogólna idea polega na wyznaczeniu maksymalnie n wektorów wspierających, pozwalających wyznaczyć hiperpłaszczyznę rozdzielającą zgodną z definicją 1. Niewątpliwą zaletą tej metody są następujące jej własności. Rozwiązanie poszukiwane jest w przestrzeni $P^* = E^{n+1}$. W przeciwieństwie do metody Ho-Kashyap znane jest oszacowanie górne maksymalnej liczby kroków, niezbędnych do stwierdzenia, że zbiory nie są liniowo rozdzielne. Nie jest więc potrzebne niepraktyczne założenie, że zbiory są rozdzielne.

Własności każdej z przedstawionych w tym rozdziele metod nie są wystarczające, aby było możliwe bezpośrednie zastosowanie ich do konstrukcji hierarchicznego klasyfikatora binarnego. Biorąc pod uwagę własności algorytmu Jóźwika, jak również fakt, że jest to podejście mało eksplorowane, został on, pomimo swoich wad, wybrany na algorytm bazowy. Jakie więc są jego wady, które uniemożliwiają użycie go w oryginalnej formie do konstrukcji klasyfikatora odcinkowo-liniowego?

Rozdział 2

Algorytm bazowy i jego własności

Wśród wszystkich przedstawionych w rozdziale pierwszym metod rozdzielania dwóch zbiorów na uwagę zasługuje algorytm Jóźwika [25], który stał się podstawą prowadzonych badań i w dalszych częściach pracy nazywany będzie algorytmem LS2S (*the algorithm of Linear Separability of two Sets*). Jest on interesujący z następujących powodów. Przede wszystkich nie wymaga informacji *a priori*, czy zbiory są rozdzielne liniowo.

Po drugie jest to *algorytm addytywny*. Wyznaczone przez algorytm rozwiązanie $\boldsymbol{a}$, które zapewnia rozdzielność elementów zbioru uczącego T, może być punktem startowym ponownego uruchomienia algorytmu dla zbioru $T^* \supset T$, przyspieszając tym samym uzyskanie rozwiązania dla zbioru uczącego T^*.

W przypadku, gdy zbiory są rozdzielne liniowo, analiza algorytmu pozwala stwierdzić, że kolejne, otrzymane na tym samym poziomie rekurencyjnych wywołań algorytmu, rozwiązania $\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_i$, potraktowane kolejno jako reguła decyzyjna projektowanego klasyfikatora, gwarantują, że klasyfikator ten poprawnie klasyfikuje elementy odpowiednich podzbiorów $T_0, T_1, \ldots, T_i$ zbioru uczącego T. Podzbiory te są w następującej relacji względem siebie: $T_0 \supset T_1 \supset T_i \supseteq T$. Innymi słowy, rozwiązania te zapewniają poprawną klasyfikację coraz większej liczby elementów zbioru uczącego. Cecha ta pozwala na bieżąco, podczas działania implementacji tego algorytmu, określić maksymalną liczbę kroków, które należy jeszcze wykonać w celu otrzymania rozwiązania poprawnie klasyfikującego cały zbiór uczący T.

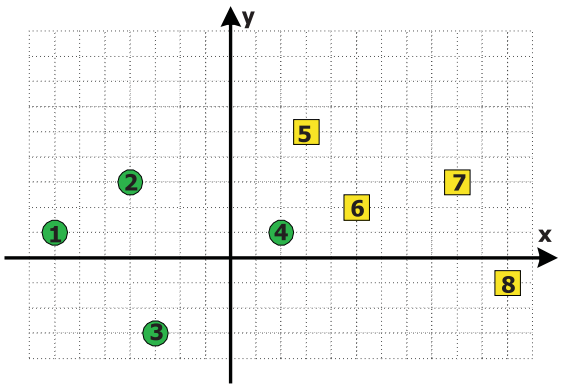
Dodatkowo, przedstawiony w pracy [27] dowód poprawności tego algorytmu, pozwala uniknąć spekulacji, że algorytm działa poprawnie dla wąskiej grupy zbiorów uczących o pewnych określonych cechach. Przyjęte w rozprawie ograniczenie ilustrowania własności wszystkich przedstawianych algorytmów dla obiektów opisanych dwoma cechami, wynika tylko i wyłącznie z potrzeby podania przejrzystych przykładów i może być swobodnie uogólnione dla obiektów opisanych przez n cech.

W przypadku, gdy zbiory są rozdzielne liniowo, istnieje nieskończenie wiele hiperpłaszczyzn rozdzielających. Liczba możliwych rozwiązań uzyskanych po uruchomieniu algorytmu LS2S jest ograniczona do tych, które opisują hiperpłaszczyzny rozpięte na elementach zawartych w rozdzielanych zbiorach. Z jednej strony stanowi to ogromną zaletę, z drugiej jednak strony — pomimo ograniczenia liczby możliwych rozwiązań — prowadzić to może do opisanej poniżej niekorzystej sytuacji.

Główną wadą algorytmu LS2S jest to, że zwracane rozwiązanie nie musi zapewniać ścisłej rozdzielności zbiorów, choć jest możliwe uzyskanie takiego rozwiązania. Oznacza to, że skończona

liczba elementów zbioru uczącego nie będzie poprawnie klasyfikowana — maksymalnie $\frac{n}{2}$ dla obiektów reprezentowanych przez n-wymiarowy wektor cech.

Ponieważ w rozprawie kluczowym elementem jest określenie algorytmu ścisłej rozdzielności dwóch zbiorów z zachowaniem wszystkich zalet opisanego powyżej algorytmu i usunięciu jego wad, poniżej przedstawiono algorytm LS2S wraz z przykładem ilustrującym wymienione powyżej cechy. W przedstawionych poniżej przykładach zakładamy, że rozdzielane zbiory X_1, X_2 składają się z przedstawionych na rysunku 2.1 punktów należących do płaszczyzny.



$$X_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$$
$$X_2 = \{\boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8\}$$

$x_1{=}[-7, 1]$ $\quad x_2{=}[-4, 3]$
$x_3{=}[-3, -3]$ $\quad x_4{=}[2, 1]$
$x_5{=}[3, 5]$ $\quad x_6{=}[5, 2]$
$x_7{=}[9, 3]$ $\quad x_8{=}[11, -1]$

Rysunek 2.1: Rozdzielane zbiory X_1 i X_2

2.1. Algorytm badania rozdzielności liniowej — LS2S

W rozdziale pierwszym do zapisu warunków (1.1), dotyczących definicji rozdzielności dwóch zbiorów X_1 i X_2, w postaci jednego warunku, stosowano przekształcenie (1.8). W związku z tym, że jest to przekształcenie wykorzystywane w algorytmie LS2S, przytoczmy je jeszcze raz:

$$\boldsymbol{t} = f(\boldsymbol{x}), \quad \boldsymbol{x} \in X = X_1 \cup X_2$$

$$f(\boldsymbol{x}) = \begin{cases} [x_1, x_2, \ldots x_n, 1] & \text{gdy } \boldsymbol{x} \in X_1 \\ [-x_1, -x_2, \cdots - x_n, -1] & \text{gdy } \boldsymbol{x} \in X_2 \end{cases} \tag{2.1}$$

Należy zwrócić uwagę, że choć po zastosowaniu tej transformacji mamy do czynienia tylko z jednym zbiorem T, nie oznacza to utraty informacji o przynależności każdego z jego elementów do jednej z dwóch klas. Pierwotna przynależność do zbioru X_1 lub X_2 może być łatwo uzyskana na podstawie wartości $(n{+}1)$-ej składowej każdego wektora $\boldsymbol{t}{\in}T$.

Po zastosowaniu przekształcenia (2.1) warunki (1.1) równoważne są:

$$\langle \boldsymbol{a}, \boldsymbol{t} \rangle \geqslant 0, \quad \boldsymbol{t} \in T = \{\boldsymbol{x} \in X_1 \cup X_2 : f(\boldsymbol{x})\} \tag{2.2}$$

gdzie $\langle ., . \rangle$ oznacza iloczyn skalarny.

Warunek (2.2) ma następującą interpretację geometryczną. Wektor $\boldsymbol{a}$ w przestrzeni E^{n+1} jest wektorem normalnym do hiperpłaszczyzny $\boldsymbol{H}$ przechodzącej przez punkt $\mathcal{O}{=}(0, \ldots, 0) \in \mathrm{E}^{n+1}$, takiej że wszystkie punkty odpowiadające klasyfikowanym obiektom, po przekształceniu (2.1) znajdują się w tej samej części przestrzeni E^{n+1} — powstałej w wyniku podziału przestrzeni E^{n+1} przez hiperpłaszczyznę $\boldsymbol{H}$.

Informacje te są wystarczające, by móc przytoczyć algorytm LS2S przedstawiony w pracy [26]. W przedstawionym algorytmie zachowano zgodność merytoryczną z algorytmem oryginalnym, choć wprowadzono drobne modyfikacje, które czynią algorytm czytelniejszym. I tak, algorytm wymaga podania dwóch parametrów wejściowych:

- T_p — lista punktów, które mają być rozdzielone,
- K_p — zbiór punktów, przez które ma przechodzić hiperpłaszyzna rozdzielająca.

Na starcie algorytmu $p=0$. Wszystkie punkty należące do zbioru uczącego mają być rozdzielone, a więc $T_0=\{f(\boldsymbol{x})\in E^{n+1}:\ \boldsymbol{x}\in X_1\cup X_2\in E^n\}$. Wobec faktu, że nie są znane żadne punkty, przez które ma przechodzić hiperpłaszczyzna rozdzielająca, drugi parametr przyjmuje wartość $K_0=\emptyset$.

W oryginalnej wersji algorytmu drugi parametr określał przestrzeń, w której poszukiwane jest rozwiązanie, i wartość początkowa tego parametru była równa $S_0=E^{n+1}$. Aby zachować zgodność z oryginałem dodano, jako pierwszy krok algorytmu, krok wyznaczający tą przestrzeń.

Jako wynik wykonania algorytmu zwracane są dwa parametry: informacja o rozdzielności zbiorów, gdy $flag=-1$ zbiory są nierozdzielne liniowo, gdy $flag=1$ zbiory są rozdzielne. W przypadku, gdy zbiory są rozdzielne liniowo drugi zwracany parametr określa wektor normalny do hiperpłaszczyzny rozdzielającej.

W przytaczanym algorytmie zmodyfikujemy punkt drugi algorytmu, tak, aby sprecyzować wybór wektora ze zbioru T_p. Jest to jedyny krok, którego odpowiednie „dodefiniowanie" umożliwia pokazanie, że tylko i wyłącznie od tego kroku i uporządkowania zbioru T_0 zależy otrzymane rozwiązanie w przypadku, gdy zbiory są rozdzielne liniowo.

Litera p używana w nazwach parametrów i sposobie numerowania kroków algorytmu może być pominięta, ale ze względu na fakt, że jest to algorytm rekursywny, umożliwia zorientowanie się o poziomie zagłębień rekurencyjnych wywołań w każdym kroku algorytmu — wartość p określa aktualny poziom. Dodatkowo wartość wyrażenia $n-p$ informuje, ile jeszcze rekurencyjnych wywołań w głąb może być wykonanych.

Nagłówek algorytmu LS2S: $(\boldsymbol{a}, flag) \leftarrow \mathrm{LS2S}(T_p\,,\ K_p)$

Algorytm LS2S:

p.1 $S_p=E^{n+1}\cap K_p^{\perp}$ — przestrzeń, w której poszukiwane jest rozwiązanie;

p.2 $\boldsymbol{a}$ — dowolny niezerowy i ortogonalny rzut wektora z listy T_p na podprzestrzeń S_p, jeśli taki rzut istnieje. *W przypadku, gdy istnieją rzuty kilku wektorów z listy T_p o podanych własnościach, wybieramy rzut pierwszego wektora o takich własnościach znajdującego się na liście T_p*;

p.3 jeśli w T_p nie istnieje żaden wektor, którego ortogonalny rzut na podprzestrzeń S_p byłby niezerowy, to przyjmij $flag=-1$ i idź do $p.15$;

p.4 $ZLE_p=\{\boldsymbol{t}\in T_p : \langle \boldsymbol{a},\boldsymbol{t}\rangle < 0\}$ — lista niepoprawnie rozdzielonych punktów;

p.5 jeśli $\mathrm{card}(ZLE_p) > \frac{\mathrm{card}(T_p)}{2}$ wówczas zmień kierunek wektora $\boldsymbol{a}$ na przeciwny $(\boldsymbol{a}\leftarrow -\boldsymbol{a})$ i wyznacz ponownie listę niepoprawnie rozdzielonych punktów tzn.:

$$ZLE_p=\{\boldsymbol{t}\in T_p : \langle \boldsymbol{a},\boldsymbol{t}\rangle < 0\}$$

p.6 jeśli $ZLE_p = \emptyset$, to ustaw $flag = 1$ i idź do kroku $p.15$ — wszystkie punkty znajdujące się na liście T_p zostały rozdzielone poprawnie;

p.7 jeśli $p{=}n$ i $ZLE_p{\neq}\emptyset, ustawflag = -1$ i idź do kroku $p.14$, — nie jest możliwe określenie hiperpłaszczyzny rozdzielającej;

p.8 $T_{p+1} = T_p \setminus ZLE_p$;

p.9 $\boldsymbol{b}$ — w oryginale: dowolny wektor z listy ZLE_p — tu: pierwszy wektor z listy ZLE_p;

p.10 $K_{p+1} = K_p \cup \{\boldsymbol{b}\}$;

p.11 wywołaj rekurencyjnie algorytm: $(\boldsymbol{a}, flag) \leftarrow$ LS2S (T_{p+1}, K_{p+1});

p.12 jeśli $flag = -1$, to idź do kroku $p.15$;

p.13 $ZLE_p = \{\boldsymbol{t} \in ZLE_p : \langle \boldsymbol{a}, \boldsymbol{t} \rangle < 0\}$;

p.14 jeśli $ZLE_p \neq \emptyset$, to idź do kroku $p.8$;

p.15 jeśli $flag{=}-1$, to zbiory nie są liniowo rozdzielne; jeśli natomiast $flag{=}1$ i $p{=}0$ wówczas zbiory są liniowo rozdzielne, a wektor $\boldsymbol{a}$ jest poszukiwanym wektorem normalnym do hiperpłaszczyzny rozdzielającej. Zwróć parametr $flag$ i wektor $\boldsymbol{a}$.

2.2. Przykłady ilustrujące własności algorytmu LS2S

Poniżej przedstawiono dwa przykłady, które ilustrują wymienione własności algorytmu bazowego LS2S. Pierwszy przykład jest dokładnym zapisem wszystkich wykonywanych kroków podczas uruchomienia algorytmu LS2S dla listy uporządkowanej w następujący sposób:

$$T_0{=}\{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4, \boldsymbol{t}_5, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\}{=}\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), f(\boldsymbol{x}_3), f(\boldsymbol{x}_4), f(\boldsymbol{x}_5), f(\boldsymbol{x}_6), f(\boldsymbol{x}_7), f(\boldsymbol{x}_8)\}$$

Oznacza to, że algorytm uruchomiony został w następujący sposób: LS2S(T_0, $\emptyset$) a rozwiązanie jest poszukiwane w przestrzeni E^3. W kroku $p.2$ algorytmu w celu wyznaczenia wektora $\boldsymbol{a}$, wyznaczającego jednoznacznie hiperpłaszczyznę rozdzielającą, wykorzystano ortogonalizację Grama-Schmidta. Przykład ten jest cenny, ponieważ pomimo swojej prostoty (w sensie łącznej liczby kroków) ilustruje rekursywną naturę algorytmu. Jednocześnie potwierdza sens wprowadzonego sposobu numerowania kroków algorytmu — co pozwala na szybkie zidentyfikowanie aktualnej głębokości wywołań rekursywnych.

0.1 $S_0 = E^3$

0.2 $\boldsymbol{a} = \boldsymbol{t}_1$

0.3 warunek niespełniony

0.4 $ZLE_0 = \{\boldsymbol{t}_4\}$

0.5 warunek niespełniony

0.6 warunek niespełniony

0.7 warunek niespełniony

0.8 $T_1 = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_5, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\}$

0.9 $\boldsymbol{b} = \boldsymbol{t}_4$

0.10 $K_1 = \emptyset \cup \{\boldsymbol{t}_4\} = \{\boldsymbol{t}_4\}$

0.11 wykonaj algorytm $(\boldsymbol{a}, flag) \leftarrow$ LS2S (T_1, K_1)

1.1 $S_1 = E^3 \cap \{\boldsymbol{t}_4\}^{\perp}$

1.2 $\boldsymbol{a} = \boldsymbol{t}_1 - \frac{\langle \boldsymbol{t}_4, \boldsymbol{t}_1 \rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4 \rangle} \boldsymbol{t}_4 = 3 \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$

1.3 warunek niespełniony

1.4 $ZLE_1 = \{\boldsymbol{t}_5\}$

1.5 warunek niespełniony

1.6 warunek niespełniony

1.7 warunek niespełniony

1.8 $T_2 = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\}$

1.9 $\boldsymbol{b} = \boldsymbol{t}_5$

1.10 $K_2 = K_1 \cup \{\boldsymbol{t}_5\} = \{\boldsymbol{t}_4, \boldsymbol{t}_5\}$

1.11 wykonaj algorytm $(\boldsymbol{a}, flag) \leftarrow$ LS2S (Z_2, K_2)

2.1 $S_2 = E^3 \cap \{\boldsymbol{t}_4, \boldsymbol{t}_5\}^{\perp}$

2.2 $\boldsymbol{a} = \boldsymbol{t}_1 - \frac{\langle \boldsymbol{t}_4, \boldsymbol{t}_1 \rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4 \rangle} \boldsymbol{t}_4 - \frac{\langle \boldsymbol{p}, \boldsymbol{t}_1 \rangle}{\langle \boldsymbol{p}, \boldsymbol{p} \rangle} \boldsymbol{p} = \frac{6}{11} \begin{bmatrix} -4 \\ 1 \\ 7 \end{bmatrix}$

gdzie $\boldsymbol{p} = \boldsymbol{t}_5 - \frac{\langle \boldsymbol{t}_4, \boldsymbol{t}_5 \rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4 \rangle} \boldsymbol{t}_4 = \begin{bmatrix} 1 \\ -3 \\ 1 \end{bmatrix}$

2.3 warunek niespełniony

2.4 $ZLE_2 = \emptyset$

2.5 warunek niespełniony

2.6 warunek spełniony

2.15 warunek niespełniony

1.12 warunek niespełniony

1.13 $ZLE_1 = \emptyset$

1.14 warunek niespełniony

1.15 warunek niespełniony

0.12 warunek niespełniony

0.13 $ZLE_0 = \emptyset$

0.14 warunek niespełniony

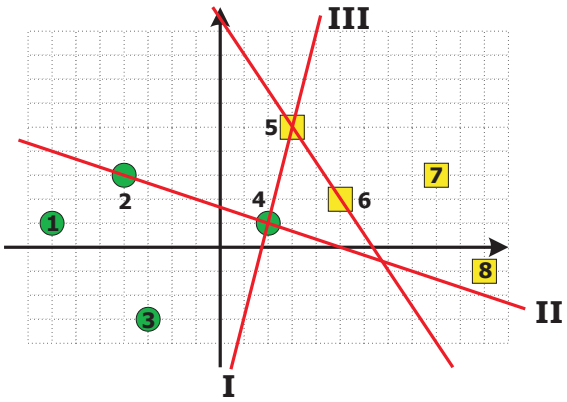
0.15 warunek spełniony i wektor $\boldsymbol{a}$ jednoznacznie określa poszukiwane rozwiązanie.

Zwróćmy uwagę, że z jednej strony otrzymane rozwiązanie $\boldsymbol{a}$ jest, w przestrzeni E^3, wektorem normalnym do płaszczyzny rozdzielającej punkty z listy T_0 otrzymane po zastosowaniu transformacji (2.1). Płaszczyzna ta przechodzi przez środek układu współrzędnych $\mathcal{O}=(0,0,0)$. Z drugiej zaś wektor $\boldsymbol{a}$ określa prostą rozdzielającą punkty ze zbiorów X_1, X_2 w przestrzeni oryginalnej.

Drugi przykład przedstawiono w postaci schematu, ilustrując ponownie rekursywną naturę algorytmu (rysunek 2.3). Dla każdego wywołania algorytmu zaprezentowano na schemacie parametry z jakimi został on wywołany. Każde z pośrednich rozwiązań przedstawiono w przestrzeni oryginalnej. Schematyczne ujęcie wykonania algorytmu, pozwala na jego ogarnięcie jako całości, ponieważ mieści się na jednej stronie. Przykład ten ilustruje, że w przypadku zbiorów rozdzielnych liniowo, kolejne otrzymane podczas działania algorytmu — w obrębie jednego wywołania — rozwiązania, zapewniają prawidłowe rozdzielenie coraz większej liczby punktów ze zbioru uczącego.

W obu przedstawionych przykładach wyznaczona przez algorytm prosta rozdzielająca przechodzi przez punkty $\boldsymbol{x}_4$ i $\boldsymbol{x}_5$. Klasyfikator wykorzystujący tę prostą umożliwiałby poprawną klasyfikację pozostałych punktów należących do zbioru uczącego. Nie byłoby jednak możliwe poprawne klasyfikowanie obu punktów $\boldsymbol{x}_4$ i $\boldsymbol{x}_5$.

Te proste przykłady ilustrują wspomnianą już wadę algorytmu LS2S. Choć jest możliwe wyznaczenie rozwiązania rozpiętego na punktach należących do jednej klasy, algorytm nie zapewnia, że takie rozwiązanie zostanie zwrócone jako wynik wykonania algorytm. Wszystkie możliwe dla podanego przykładu rozwiązania przedstawiono na rysunku 2.2.



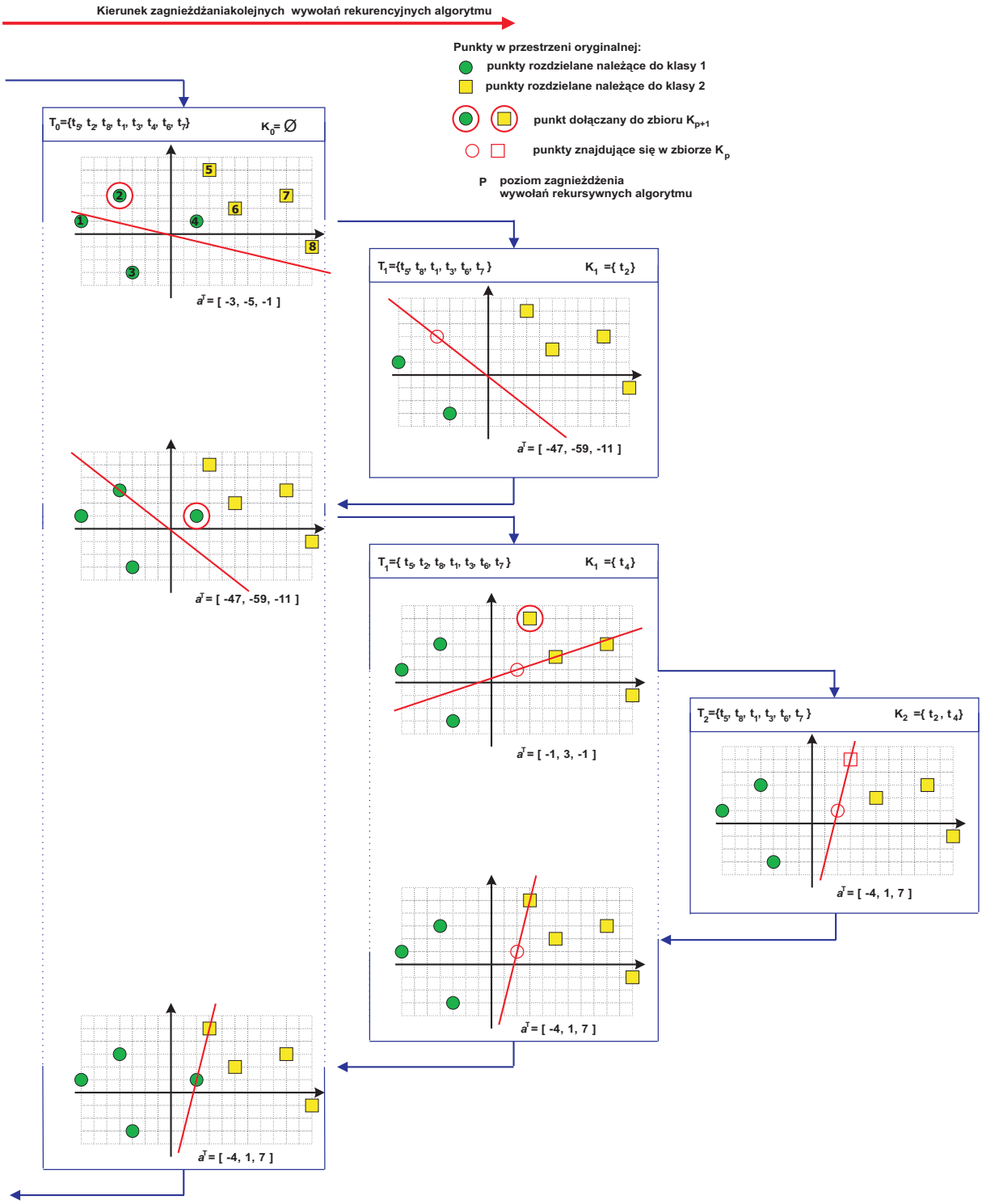
Rysunek 2.2: Możliwe rozwiązania zwracane przez oryginalny algorytm LS2S

Od czego więc zależy, które rozwiązanie wśród wszystkich możliwych otrzymamy jako wynik wykonania algorytmu? Zależy to od dwóch czynników. Pierwszym jest sposób, w jaki „dookreślimy" na potrzeby maszynowej implementacji krok $p.2$ algorytmu LS2S. Drugim jest określenie sposobu wybierania jednego punktu z wszystkich niepoprawnie rozdzielonych punktów w kroku $p.8$. Z tego właśnie powodu w definicji algorytmu parametry T_p i ZLE_p są listami a nie zbiorami, a wszystko to po to, aby wybór niepoprawnie rozdzielonego punktu mógł być jednoznacznie określony.

Dla przyjętej definicji kroku $p.2$ nie jest możliwe otrzymanie rozwiązania (III), natomiast rozwiązanie (II) uzyskamy dla następującego uporządkowania:

$$T_0 = \{f(\boldsymbol{x}_3), f(\boldsymbol{x}_2), f(\boldsymbol{x}_1), f(\boldsymbol{x}_4), f(\boldsymbol{x}_5), f(\boldsymbol{x}_6), f(\boldsymbol{x}_7), f(\boldsymbol{x}_8)\}\,.$$

Naturalnym wydaje się więc zadanie następującego pytania. Czy jest możliwe zaproponowanie algorytmu rozdzielającego dwa zbiory, który będzie charakteryzował się wszystkimi zaletami przedstawionego algorytmu LS2S, z jednoczesnym usunięciem jego, niewygodnej z punktu widzenia zastosowań, wady?



Rysunek 2.3: LS2S dla uporządkowania: $\{\boldsymbol{x}_5, \boldsymbol{x}_2, \boldsymbol{x}_8, \boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_6, \boldsymbol{x}_7\}$

Rozdział 3

Algorytmy rozdzielające z prześwitem ε

W rozdziale tym uwagę skupimy na algorytmach umożliwiających rozdzielenie dwóch zbiorów z prześwitem, jeśli jest to możliwe, co zapewniałoby pozbycie się przedstawionej w poprzednim rozdziale wady algorytmu bazowego.

Z punktu widzenia nadrzędnego celu pracy, jakim jest konstrukcja klasyfikatora odcinkowo-liniowego, niezbędnym krokiem było określenie algorytmu rozdzielającego zbiory z maksymalnym prześwitem. W początkowej fazie prac korzystnym wydawało się, podzielenie pracy, związanej z tym zagadnieniem, na dwa etapy. Pierwszy, obejmować miał definicję algorytmu badającego rozdzielność dwóch zbiorów z określoną wartością prześwitu ε. Doświadczenie zebrane w tym etapie miało posłużyć, do określenia algorytmu badającego rozdzielność dwóch zbiorów z maksymalnym prześwitem. Kolejność prac wynikała z przeświadczenia, że drugi etap wydawał się dużo trudniejszy, ze względu na element „optymalizacji" szerokości prześwitu ε. Tymczasem wbrew intuicji, okazało się, że etap pierwszy doprowadził autorkę do zaskakujących konkluzji. Podczas realizacji etapu drugiego, okazało się, że wykorzystanie pewnej informacji powoduje, że przestajemy mieć do czynienia z zagadnieniem optymalizacyjnym, gdyż w efekcie mamy do czynienia z podaniem warunków istnienia (i wyznaczeniem) jedynego, niezerowego rozwiązania na maksymalną wielkość prześwitu ε.

Rozdział kończy podanie definicji algorytmu SLS2S (*the algorithm of Strict and maximum Linear Separability of two Sets*), który umożliwia rozdzielenie dwóch zbiorów, jeśli jest to możliwe, w optymalny sposób, czyli z maksymalnym prześwitem. Podobnie jak w rozdziale drugim podano również przykłady wykonania przedstawionego algorytmu ilustrując różnice między algorytmem SLS2S w stosunku do przedstawionego w rozdziale 2 — LS2S.

W przypadku algorytmów rozdzielania dwóch zbiorów z prześwitem i z maksymalnym prześwitem warunki opisujące parę hiperpłaszczyzn (1.5 i 1.6) możemy zapisać w postaci jednej nierówności po zastosowaniu poniższej transformacji:

$$\begin{gathered} \boldsymbol{y} = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in X = X_1 \cup X_2 \\ f^*(\boldsymbol{x}) = \begin{cases} [x_1, x_2, \dots x_n, 0, 1] & \text{gdzie } \boldsymbol{x} \in X_1 \\ [-x_1, -x_2, \cdots - x_n, -1, -1] & \text{gdzie } \boldsymbol{x} \in X_2 \end{cases} \end{gathered} \tag{3.1}$$

Odpowiednikiem wspomnianych warunków są wówczas:

$$\langle \boldsymbol{a}^*, \boldsymbol{y} \rangle \geqslant 0 \quad \boldsymbol{y} \in Y = \{\boldsymbol{x} \in X_1 \cup X_2 : f^*(\boldsymbol{x})\} \tag{3.2}$$

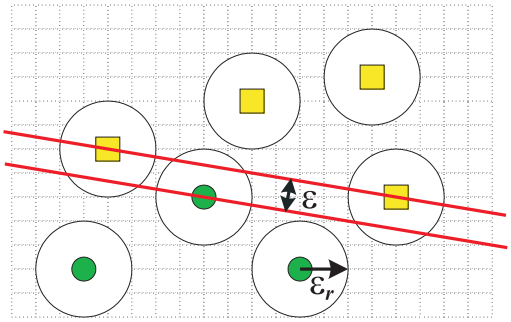
gdzie $\langle ., . \rangle$ oznacza iloczyn skalarny, zaś $\boldsymbol{a}^* = [a_1, a_2, \dots a_n, \varepsilon, a_{n+2}] \in \mathrm{E}^{n+2}$ i jest to wektor o następującej własności: $\sum_{i=1}^{n} (a_i^*)^2 = 1$.

Podkreślmy, że rozwiązania $\boldsymbol{a}^*$ poszukujemy wówczas w przestrzeni E^{n+2}, a nie w przestrzeni E^n i pozwala ono w jednoznaczny sposób określić hiperpłaszczyzny rozdzielające h_1 i h_2 w przestrzeni oryginalnej.

3.1. Algorytm rozdzielania zbiorów z prześwitem ε

Przekształcenie (3.2) umożliwia sprowadzenie zagadnienia znajdowania rozwiązania $\boldsymbol{a}^*$ do znalezienia tego rozwiązania przy pomocy zmodyfikowanej wersji algorytmu LS2S z uwzględnieniem dodatkowego warunku dotyczącym wartości $(n+1)$-wszej składowej tego wektora.

Określając algorytm rozdzielania zbiorów z prześwitem, który ma mieć wszystkie zalety algorytmu LS2S włącznie ze strukturą, należy zadbać o to, by wszystkie kroki tego algorytmu były możliwe do wykonania. Newralgicznym jest odpowiednik kroku $p.2$ algorytmu LS2S. Pojawia się bowiem pytanie: jakie własności muszą spełniać punkty należące do rozdzielanych zbiorów X_1 i X_2, aby było możliwe wyliczenie wektora $\boldsymbol{a}^*$, którego $(n+1)$-wsza składowa jest równa ε? Odpowiedź na to pytanie niestety nie jest prosta. Stwierdzenie: „Punkty powinny być oddalone od siebie w przestrzeni oryginalnej o co najmniej ε," jest tylko pozornie prawdziwe. Fałszywość tej tezy zilustrowano na rysunku 3.1, gdzie wszystkie punkty oddalone są od siebie o więcej niż ε_r, ale nie oznacza to, że zbiory są rozdzielne z $\varepsilon = \varepsilon_r$.



Rysunek 3.1: Pozorna zależność odległości między punktami a szerokością prześwitu

Z drugiej strony, wykorzystanie algorytmu LS2S z dodatkowym ograniczeniem, oznacza że pozostaje w mocy teoretyczna podstawa tego algorytmu [25] zapisana w postaci następującego twierdzenia.

Twierdzenie 1
Załóżmy, że istnieje rozwiązanie dla p punktów znajdujących się na liście Y, tzn.:

$$\forall_{1 \leqslant j \leqslant p} \langle \boldsymbol{a}^{**}, \boldsymbol{y}_j \rangle \geqslant 0$$

Wyznaczone do tej pory rozwiązanie rozdziela $(i-1)$ pierwszych punktów $(i < p)$, tzn:

$$\forall_{1 \leqslant j < i} \left\langle \boldsymbol{a}^{(i-1)}, \boldsymbol{y}_j \right\rangle \geqslant 0$$

Wektory $\boldsymbol{a}^{(i-1)}$ i $\boldsymbol{a}^{**}$ nie są współliniowe. Rozwiązanie $\boldsymbol{a}^{(i-1)}$ choć rozdziela prawidłowo $(i-1)$ pierwszych punktów, to nie zapewnia rozdzielności dla i-tego punktu, tzn:

$$\left\langle \boldsymbol{a}^{(i-1)}, \boldsymbol{y}_i \right\rangle < 0$$

Wówczas istnieje rozwiązanie zapewniające rozdzielność pierwszych i *punktów:*

$$\boldsymbol{a}^{(i)} = \frac{\left\langle \boldsymbol{a}^{(i-1)}, \boldsymbol{y}_i \right\rangle \boldsymbol{a}^{**} - \langle \boldsymbol{a}^{**}, \boldsymbol{y}_i \rangle \, \boldsymbol{a}^{(i-1)}}{\left\langle \boldsymbol{a}^{(i-1)} - \boldsymbol{a}^{**}, \boldsymbol{y}_i \right\rangle}$$

Twierdzenie 1 uzasadnia umieszczenie punktu źle sklasyfikowanego w zbiorze punktów, przez które ma przechodzić odpowiednia hiperłaszczyzna rozdzielająca, gdyż $\left\langle \boldsymbol{a}^{(i)}, \boldsymbol{y}_j \right\rangle = 0$.

Załóżmy, że możliwe jest, aby każde pośrednie rozwiązanie $\boldsymbol{a}^{(i-1)}$ oraz nieznane, ale istniejące rozwiązanie $\boldsymbol{a}^{**}$ dla listy Y, miało własność wynikającą z narzuconych ograniczeń, czyli aby ich składowa $(n+1)$ była równa ε. Wówczas przyjmując następujące oznaczenia:

$$\delta = \frac{\left\langle \boldsymbol{a}^{(i-1)}, \boldsymbol{y}_i \right\rangle}{\left\langle \boldsymbol{a}^{(i-1)} - \boldsymbol{a}^{**}, \boldsymbol{y}_i \right\rangle}, \quad \gamma = \frac{-\langle \boldsymbol{a}^{**}, \boldsymbol{y}_i \rangle}{\left\langle \boldsymbol{a}^{(i-1)} - \boldsymbol{a}^{**}, \boldsymbol{y}_i \right\rangle} \tag{3.3}$$

istniejące według twierdzenia 1 rozwiązanie $\boldsymbol{a}^{(i)}$, równe jest:

$$\boldsymbol{a}^{(i)} = \begin{bmatrix} a_1^{(i)} \\ \vdots \\ a_n^{(i)} \\ a_{n+1}^{(i)} \\ a_{n+2}^{(i)} \end{bmatrix} = \delta \begin{bmatrix} a_1^{**} \\ \vdots \\ a_n^{**} \\ \varepsilon \\ a_{n+2}^{**} \end{bmatrix} + \gamma \begin{bmatrix} a_1^{(i-1)} \\ \vdots \\ a_n^{(i-1)} \\ \varepsilon \\ a_{n+2}^{(i-1)} \end{bmatrix} \tag{3.4}$$

Po znormalizowaniu względem pierwszych n składowych:

$$\hat{\boldsymbol{a}}^{(i)} = \frac{\boldsymbol{a}^{(i)}}{d}$$

gdzie:

$$\begin{aligned} d^2 &= \sum_{j=1}^{n} \left(a_j^{(i)}\right)^2 = \sum_{j=1}^{n} \left(\delta a_j^{**} + \gamma a_j^{(i-1)}\right)^2 = \\ &= \sum_{j=1}^{n} \left(\delta^2 \left(a_j^{**}\right)^2 + \gamma^2 \left(a_j^{(i-1)}\right)^2\right) + 2\delta\gamma \sum_{j=1}^{n} a_j^{**} a_j^{(i-1)} \overset{(1)}{=} \\ &\overset{(1)}{=} \delta^2 + \gamma^2 + 2\delta\gamma \textstyle\sum_{j=1}^{n} a_j^{**} a_j^{(i-1)} = \\ &= (\delta+\gamma)^2 + 2\delta\gamma \textstyle\sum_{j=1}^{n} \left(a_j^{**} a_j^{(i-1)} - 1\right) \overset{(2)}{<} (\delta+\gamma)^2 \end{aligned}$$

uwzględniając:

(1) $\sum_{j=1}^{n} \left(a_j^{**}\right)^2 = \sum_{j=1}^{n} \left(a_j^{(i-1)}\right)^2 = 1$ normalizacja n pierwszych składowych;

(2) $-1 < \sum_{j=1}^{n} a_j^{**} a_j^{(i-1)} < 1$ wektory $\boldsymbol{a}^{(i-1)}$ i $\boldsymbol{a}^{**}$ nie są współliniowe;

otrzymane rozwiązanie zapewnia rozdzielność z większą, choć nie znaną wartością niż zakładane na początku ε:

$$\hat{\boldsymbol{a}}^{(i)} = \frac{\boldsymbol{a}^{(i)}}{d} \quad \Leftrightarrow \quad \hat{a}_{n+1}^{(i)} = \frac{\delta+\gamma}{d}\,\varepsilon > \varepsilon$$

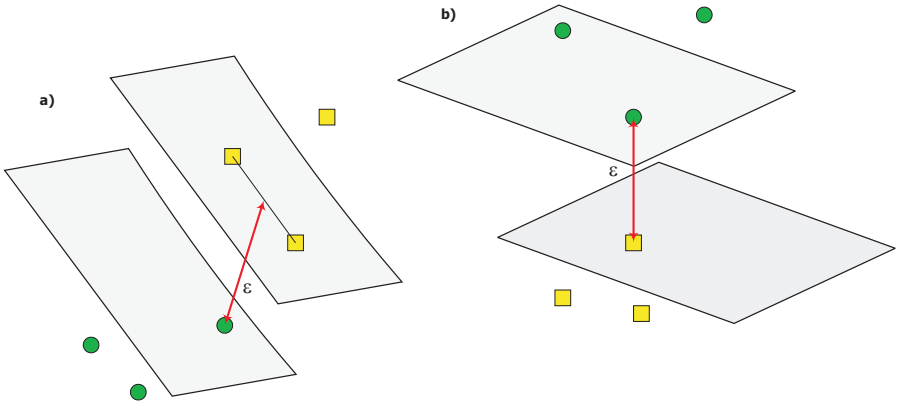
Ten zaskakujący wniosek stał się powodem przystąpienia od razu do etapu drugiego, bez podania algorytmu znajdującego wektor $\boldsymbol{a}^*$, który jednoznacznie określa dwie hiperpłaszczyzny rozdzielające odległe od siebie o konkretną, zadana wartość ε.

3.2. Algorytm rozdzielania zbiorów z maksymalnym prześwitem ε

3.2.1. Idea algorytmu

Oryginalny algorytm rekursywny [25, 27] rozpina hiperpłaszczyznę rozdzielającą na wybranych, źle sklasyfikowanych punktach ze zbiorów rozdzielanych. Aby **jednoznacznie** wyznaczyć hiperpłaszczyznę rozdzielającą w przestrzeni E^n wymagane jest podanie n punktów, które rozpinają podprzestrzeń E^{n-1}. Tymczasem w przypadku wyznaczania dwóch równoległych i maksymalnie od siebie oddalonych hiperpłaszczyzn h_1 i h_2 sytuacja wygląda zupełnie inaczej.

Spostrzeżeniem będącym podstawą algorytmu jest fakt, że do wyznaczenia hiperpłaszczyzn h_1 i h_2 wystarczy, aby znane były przynajmniej dwa punkty: jeden, przez który przechodzi hiperpłaszczyzna h_1 i drugi, przez który przechodzi hiperpłaszczyzna h_2. Ponieważ z założenia h_1 i h_2 mają być maksymalnie oddalone i nie pokrywające się, implikuje to ich równoległość. Wspomniane zaś punkty rozpinają w sposób jednoznaczny te hiperpłaszczyzny. Taką parę punktów, w dalszych rozważaniach, będziemy traktować jako *minimalną listę rozpinającą*. W przypadku tak określonej minimalnej listy punktów rozpinających, prosta łącząca te dwa punkty wyznacza kierunek normalny obu hiperpłaszczyzn h_1 i h_2. Ilustrują to przykłady w przestrzeni E^3 — rysunek 3.2a) trzy punkty na liście rozpinającej; 3.2b) minimalna, dwupunktowa lista rozpinająca.



Rysunek 3.2: Jednoznaczność „rozpinanych” hiperpłaszczyzn h_1 i h_2

W przedstawianym poniżej algorytmie SLS2S, sprawdzane jest, czy lista rozpinająca jest przynajmnniej listą minimalną (kroki $p.2$ i $p.3$). Jeśli nie, dołączane są dowolne punkty, tak aby na liście „rozpinających” punktów znajdował się przynajmniej jeden punkt z każdej klasy. Zapewnia to, że istnieje tylko jedna para hiperpłaszczyzn h_1 i h_2, które są maksymalnie od siebie oddalone. W przeciwieństwie do technik SVM [32], proponowany algorytm wyznacza odpowiedni wektor $\boldsymbol{\alpha}$ i współczynnik ε, przy spełnieniu dodatkowego warunku $\sum_{i=1}^{n} \alpha_i^2 = 1$. W technikach SVM ustalana jest wartość $\varepsilon = 1$ a jako rozwiązania poszukuje się wektora $\boldsymbol{\alpha}$ o minimalnej długości.

3.2.2. Algorytm SLS2S

Przedstawiony poniżej algorytm SLS2S jest algorytmem, który zachowuje wszystkie pozytywne cechy posiadane przez LS2S z jednoczesnym usunięciem jego wady, o której mowa była w rozdziale 2. Zachowano w miarę możliwości podobną strukturę zapisu algorytmu. W dodatku A przedstawiono formalny dowód wykonalności kroku $p.4$ algorytmu SLS2S.

Algorytm SLS2S ma następujące parametry wejściowe:

- Y_p — lista rozdzielanych punktów,
- $K_p^{(1)}$ — lista punktów, przez które ma przechodzić hiperpłaszczyzna h_1,
- $K_p^{(2)}$ — lista punktów, przez które ma przechodzić hiperpłaszczyzna h_2

i dwa parametry wyjściowe:

- $flag$ — wartość 1 oznacza, że zbiory są liniowo rozdzielne; wartość -1 oznacza, że zbiory nie są rozdzielne,
- $\boldsymbol{a}^*$ — rozwiązanie definiujące ścisłą i maksymalną rozdzielność zbiorów; wartość parametru (wektora) ma sens, gdy $flag = 1$.

Lista rozpinająca, będąca odpowiednikiem zbioru K_p w algorytmie LS2S, została podzielona na dwie listy $K_p^{(1)}$ i $K_p^{(2)}$, aby zwiększyć czytelność algorytmu i podanych w dalszej części przykładów. Na starcie algorytmu $p = 0$. Wszystkie punkty, które należy rozdzielić, tworzą listę Y_0:

$$Y_0 = \{ f^*(\boldsymbol{x}) \in E^{n+2} \colon \boldsymbol{x} \in X_1 \cup X_2 \in E^n \}.$$

Nieznane są punkty rozpinające hiperpłaszczyzny rozdzielające, więc $K_0^{(1)} = \emptyset$ i $K_0^{(2)} = \emptyset$. Dodatkowo, by zwiększyć czytelność algorytmu poprzez zmniejszenie liczby niezbędnych kroków, zdefiniowano następującą funkcję:

$$class(\boldsymbol{y}) = \begin{cases} 1 & \text{gdy } (n+2) \text{ składowa wektora } \boldsymbol{y} \text{ jest równa } 1 \\ 2 & \text{gdy } (n+2) \text{ składowa wektora } \boldsymbol{y} \text{ jest równa } -1 \end{cases} \tag{3.5}$$

Pozwala ona w sposób zwarty zapisać kroki od $p.10$ do $p.12$ algorytmu.

Parametr p w numeracji kroków algorytmu jest wprowadzony tylko i wyłącznie z punktu widzenia czytelności algorytmu. Z jednej strony oznacza on poziom rekurencyjnych wywołań algorytmu w głąb, z drugiej mówi o liczbie elementów zawartych na liście rozpinającej $K_p^{(1)} \cup K_p^{(2)}$.

Nagłówek algorytmu SLS2S: $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_p, K_p^{(1)}, K_p^{(2)})$

Algorytm SLS2S:

$p.1$ $K = \emptyset$ tzn. startowa, minimalna lista „rozpinająca";

$p.2$ jeśli card $\left(K_p^{(1)}\right) = 0$,
wówczas $K = \{$pierwszy $\boldsymbol{y} \in Y_p,$ dla którego $class(\boldsymbol{y}) = 1\}$;

$p.3$ jeśli card $\left(K_p^{(2)}\right) = 0$,
wówczas $K = K \cup \{$pierwszy $\boldsymbol{y} \in Y_p,$ dla którego $class(\boldsymbol{y}) = 2\}$;

$p.4$ $\boldsymbol{a}^*$ — wektor w przestrzeni E^{n+2} prostopadły do każdego wektora z listy „rozpinającej" $K \cup K_p^{(1)} \cup K_p^{(2)}$ z maksymalną wartością składnika ε;

p.5 $ZLE_p = \{\boldsymbol{y} \in Y_p : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$, tzn. lista źle rozdzielonych punktów;

p.6 jeśli $a^*_{n+1} = 0$ (tzn. $\varepsilon = 0$) i $\mathrm{card}(ZLE_p) > \frac{\mathrm{card}(Y_p)}{2}$ wówczas zmień kierunek wektora $\boldsymbol{a}$ na przeciwny ($\boldsymbol{a} \leftarrow -\boldsymbol{a}$) i wyznacz ponownie listę niepoprawnie rozdzielonych punktów, tzn.:

$$ZLE_p = \{\boldsymbol{y} \in Y_p : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$$

p.7 jeśli $ZLE_p = \emptyset$, ustaw $flag = 1$ i idź do kroku p.18, tzn. wszystkie punkty zostały poprawnie rozdzielone;

p.8 jeśli $p = n+1$ i $ZLE_p \neq \emptyset$, ustaw $flag = -1$ i idź do kroku p.18, tzn. zbiory nie są rozdzielne liniowo;

p.9 $Y_{p+1} = Y_p \setminus ZLE_p$;

p.10 $\boldsymbol{b}$ — pierwszy wektor z listy ZLE_p;

p.11 $c = class\,(\boldsymbol{b})$;

p.12 jeśli $\mathrm{card}\left(K_p^{(c)}\right) < n$, dodaj do odpowiedniego fragmentu listy rozpinającej $K_{p+1}^{(c)} = K_p^{(c)} \cup \{\boldsymbol{b}\}$, $K_{p+1}^{(3-c)} = K_p^{(3-c)}$

p.13 jeśli $\mathrm{card}\left(K_p^{(c)}\right) = n$, ustaw $flag = -1$ i idź do kroku p.18, tzn. zbiory nie są rozdzielne liniowo;

p.14 wywołaj rekurencyjnie algorytm $(\boldsymbol{a}^*, flag) \leftarrow \mathrm{SLS2S}\ (Y_{p+1}, K_{p+1}^{(1)}, K_{p+1}^{(2)})$;

p.15 jeśli $flag = -1$, idź do kroku p.18;

p.16 $ZLE_p = \{\boldsymbol{y} \in ZLE_p : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$;

p.17 jeśli $ZLE_p \neq \emptyset$, idź do kroku p.9;

p.18 jeśli $flag = -1$, zbiory nie są rozdzielne liniowo; w przeciwnym wypadku i dodatkowo, jeśli $p = 0$, to zbiory są rozdzielne liniowo a wektor $\boldsymbol{a}^*$ jest szukanym rozwiązaniem umożliwiającym określenie hiperpłaszczyzn rozdzielających h_1 i h_2 w sposób ścisły i maksymalny. Zwróć parametr $flag$ i wektor $\boldsymbol{a}^*$.

Teoretyczne własności algorytmu SLS2S są identyczne jak przedstawione w pracy [27] własności algorytmu LS2S. Oznacza to, że pesymistyczna złożoność algorytmu SLS2S to $\mathcal{O}(m^{n+1})$ ($m = \mathrm{card}(X_1 \cup X_2)$), zaś maksymalna liczba kroków, aby stwierdzić, że zbiory nie są rozdzielne liniowo, wynosi $\binom{m+n}{n+1}$.

3.2.3. Przykłady ilustrujace własności algorytmu SLS2S

Poniżej przedstawimy dwa przykłady, które ilustrują podstawowe własności algorytmu SLS2S. Istotne jest to, że każde z rekurencyjnych wywołań — tak jak algorytm LS2S — ma znaleźć (jeśli to możliwe) rozwiązanie dla listy dobrze rozdzielonych punktów, przy dodanym do listy rozpinającej jednym punkcie, dotychczas źle rozdzielonym.

Jeśli lista rozpinająca $K_p^{(1)} \cup K_p^{(2)}$ nie zawiera minimalnej listy rozpinającej, zostaje ona uzupełniona przez punkt $\tilde{\boldsymbol{y}}_1$ i (lub) $\tilde{\boldsymbol{y}}_2$, gdzie $\tilde{\boldsymbol{y}}_i$ jest pierwszym wektorem z listy Y_0, dla którego $class(\tilde{\boldsymbol{y}}_i) = i$.

Pierwszy z przykładów, przedstawiony krok po kroku zgodnie z definicją algorytmu, dotyczy zbioru punktów znajdujących się na rysunku 2.1. Przy czym lista punktów, które mają zostać rozdzielone jest następująca (zgodnie z przekształceniem (3.1)):

$$\begin{aligned} Y_0 &= \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_6, \boldsymbol{y}_7, \boldsymbol{y}_8\} \\ &= \{f^*(\boldsymbol{x}_4), f^*(\boldsymbol{x}_5), f^*(\boldsymbol{x}_2), f^*(\boldsymbol{x}_3), f^*(\boldsymbol{x}_1), f^*(\boldsymbol{x}_6), f^*(\boldsymbol{x}_7), f^*(\boldsymbol{x}_8)\} \end{aligned}$$

Algorytm SLS2S uruchamiany jest z następującą listą parametrów: SLS2S$(Y_0, \emptyset, \emptyset)$.

0.1 $K = \emptyset$

0.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

0.3 warunek spełniony $K = K \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_4, \boldsymbol{y}_5\}$

0.4 należy wyznaczyć **jedyne** rozwiązanie poniższego układu równań:

$$\left\{\begin{array}{rcl} \langle \boldsymbol{a}^*, \boldsymbol{x}_4 \rangle & = & 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_5 \rangle & = & 0 \\ \alpha_1^2 + \alpha_2^2 & = & 1 \\ max(\varepsilon) & & \end{array}\right. \Longrightarrow \left\{\begin{array}{rcl} 2\alpha_1 + \alpha_2 + \alpha & = & 0 \\ -3\alpha_1 - 5\alpha_2 - \varepsilon - \alpha & = & 0 \\ \alpha_1^2 + \alpha_2^2 & = & 1 \\ max(\varepsilon) & & \end{array}\right.$$

Wyznaczając z pierwszego równania α, z drugiego α_1 po podstawieniu do trzeciego równania otrzymamy:

$$17\alpha_2^2 + 8\varepsilon\alpha_2 + \varepsilon^2 - 1 = 0$$

Potraktujmy równanie jako równanie kwadratowe, gdzie ε pełni rolę parametru, wówczas:

$$\Delta_{\alpha_2} = -4\varepsilon^2 + 68$$

Spełniając trzecie równanie powyższego układu, przyjmujemy największą z możliwych do przyjęcia wartości $\varepsilon = \sqrt{17}$. W konsekwencji otrzymujemy jedyne możliwe rozwiązanie przechodzące przez punkty $\boldsymbol{x}_4, \boldsymbol{x}_5$:

$$\boldsymbol{a}^* = \frac{\sqrt{17}}{17} \begin{bmatrix} -1 \\ -4 \\ 17 \\ 6 \end{bmatrix}$$

0.5 $ZLE_0 = \{\boldsymbol{y}_2, \boldsymbol{y}_6, \boldsymbol{y}_7, \boldsymbol{y}_8\}$

0.6 warunek niespełniony

0.7 warunek niespełniony

0.8 warunek niespełniony

0.9 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

0.10 $\boldsymbol{b} = \boldsymbol{y}_2$

0.11 $c = 1$;

0.12 warunek spełniony, $K_1^{(1)} = \{\boldsymbol{y}_2\}$, $K_1^{(2)} = K_0^{(2)} = \emptyset$

0.13 warunek niespełniony

0.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_1, K_1^{(1)}, K_1^{(2)})$

1.1 $K = \emptyset$

1.2 warunek niespełniony

1.3 warunek spełniony $K = \{\boldsymbol{y}_5\}$

1.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{x}_2 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_5 \rangle &= 0 \\ \alpha_1^2 + \alpha_2^2 &= 1 \\ max(\varepsilon) \end{cases} \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{53}}{53} \begin{bmatrix} -7 \\ -2 \\ 53 \\ -22 \end{bmatrix}$$

1.5 $ZLE_1 = \{\boldsymbol{y}_4\}$

1.6 warunek niespełniony

1.7 warunek niespełniony

1.8 warunek niespełniony

1.9 $Y_2 = \{\boldsymbol{y}_5, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

1.10 $\boldsymbol{b} = \boldsymbol{y}_4$

1.11 $c = 1$;

1.12 warunek spełniony, $K_2^{(1)} = K_1^{(1)} \cup \{\boldsymbol{y}_4\} = \{\boldsymbol{y}_2, \boldsymbol{y}_4\}$,
$K_2^{(2)} = K_1^{(2)} = \emptyset$

1.13 warunek niespełniony

1.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_2, K_2^{(1)}, K_2^{(2)})$

2.1 $K = \emptyset$

2.2 warunek niespełniony

2.3 warunek spełniony $K = \{\boldsymbol{y}_5\}$

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{x}_2 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_4 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_5 \rangle &= 0 \\ \alpha_1^2 + \alpha_2^2 &= 1 \\ max(\varepsilon) \end{cases} \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{10}}{10} \begin{bmatrix} -1 \\ -3 \\ 13 \\ 5 \end{bmatrix}$$

2.5 $ZLE_2 = \emptyset$

2.6 warunek niespełniony

2.7 warunek spełniony

2.18 warunek niespełniony

1.15 warunek niespełniony

1.16 $ZLE_1 = \emptyset$

1.17 warunek niespełniony

1.18 warunek niespełniony

0.15 warunek niespełniony

0.16 $ZLE_0 = \{\boldsymbol{y}_6, \boldsymbol{y}_8\}$

0.17 warunek spełniony

0.9 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7\}$

0.10 $\boldsymbol{b} = \boldsymbol{y}_6$

0.11 $c = 2$;

0.12 warunek spełniony, $K_1^{(1)} = K_0^{(1)} = \emptyset$, $K_1^{(2)} = \{\boldsymbol{y}_6\}$

0.13 warunek niespełniony

0.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_1, K_1^{(1)}, K_1^{(2)})$

1.1 $K = \emptyset$

1.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

1.3 warunek niespełniony

1.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{x}_4 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_6 \rangle &= 0 \\ \alpha_1^2 + \alpha_2^2 &= 1 \\ max(\varepsilon) & \end{cases} \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{10}}{10} \begin{bmatrix} -3 \\ -1 \\ 10 \\ 7 \end{bmatrix}$$

1.5 $ZLE_1 = \{\boldsymbol{y}_5\}$

1.6 warunek niespełniony

1.7 warunek niespełniony

1.8 warunek niespełniony

1.9 $Y_2 = \{\boldsymbol{y}_4, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7\}$

1.10 $\boldsymbol{b} = \boldsymbol{y}_5$

1.11 $c = 2$;

1.12 warunek spełniony, $K_2^{(1)} = K_1^{(1)} = \emptyset$,
$K_2^{(2)} = K_1^{(2)} \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_5, \boldsymbol{y}_6\}$

1.13 warunek niespełniony

1.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_2, K_2^{(1)}, K_2^{(2)})$

2.1 $K = \emptyset$

2.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

2.3 warunek niespełniony

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{x}_4 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_5 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{x}_6 \rangle &= 0 \\ \alpha_1^2 + \alpha_2^2 &= 1 \\ max(\varepsilon) \end{cases} \quad \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{13}}{13} \begin{bmatrix} -3 \\ -2 \\ 11 \\ 8 \end{bmatrix}$$

2.5 $ZLE_2 = \emptyset$

2.6 warunek niespełniony

2.7 warunek spełniony

2.18 warunek niespełniony

1.15 warunek niespełniony

1.16 $ZLE_1 = \emptyset$

1.17 warunek niespełniony

1.18 warunek niespełniony

0.15 warunek niespełniony

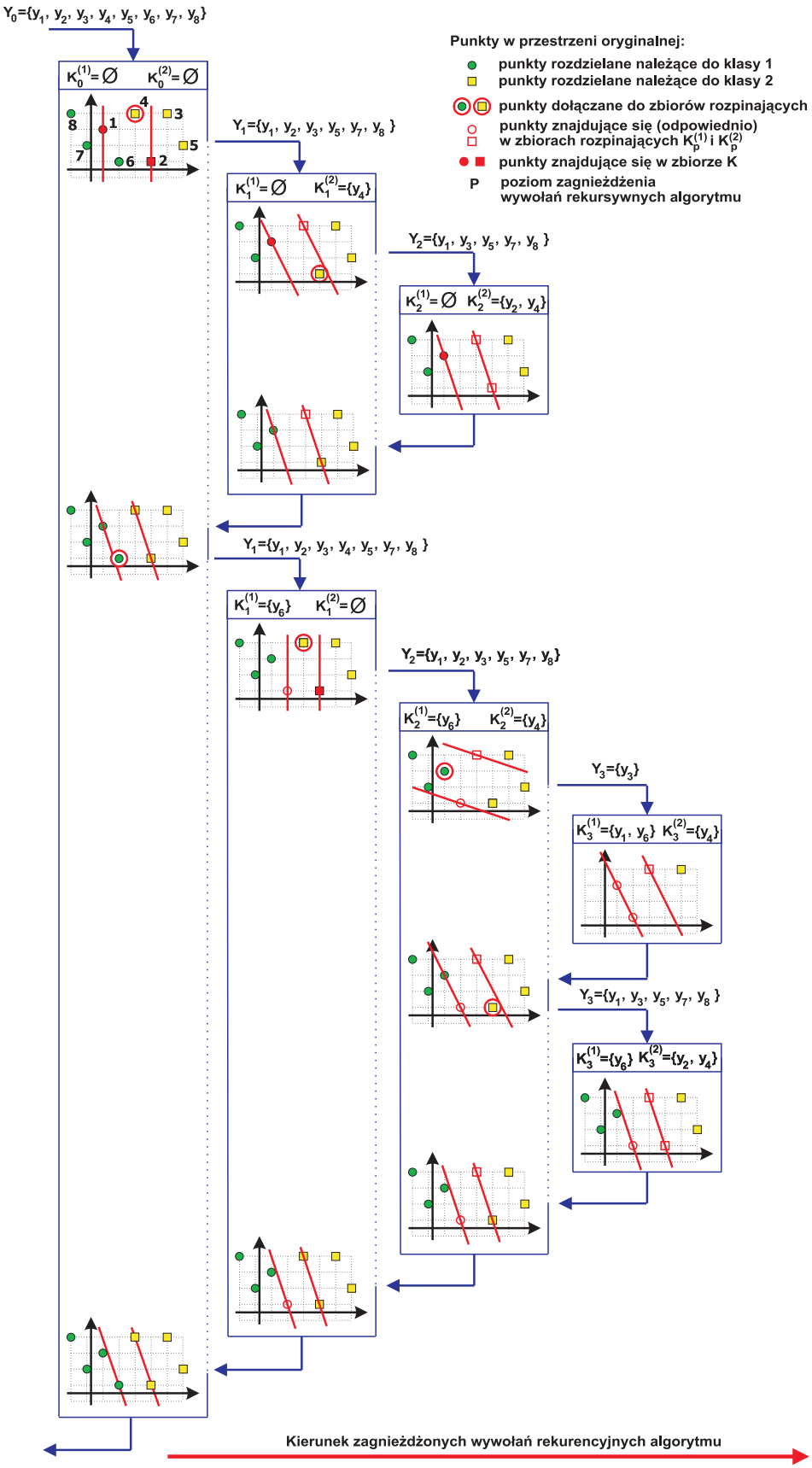
0.16 $ZLE_0 = \emptyset$

0.17 warunek niespełniony

0.18 warunek spełniony — $\boldsymbol{a}^*$ jest poszukiwanym rozwiązaniem.

Drugi przykład (rysunek 3.3) nie zawiera żadnych obliczeń i jest przedstawiony w postaci schematu, gdzie każde z uzyskiwanych pośrednich rozwiązań zostało umieszczone w przestrzeni oryginalnej.

Analiza algorytmu i podanych przykładów rodzi następujące pytania. W jaki sposób wyliczać wektor $\boldsymbol{a}^*$ w implementacji maszynowej kroku $p.4$ algorytmu SLS2S? Na ile fragmenty dowodu wykonalności kroku $p.4$ przedstawionego w dodatku A mogą być przydatne w automatycznym wyliczaniu $\boldsymbol{a}^*$? Czy są jakieś przesłanki, które pozwolą zoptymalizować algorytmu pod względem liczby wykonań kroku $p.4$?



Rysunek 3.3: Przykładowy schemat działania algorytmu SLS2S

Rozdział 4

Uwagi dotyczące implementacji algorytmu SLS2S

Osobnym zagadnieniem, dotyczącym algorytmu SLS2S, jest implementacja maszynowa jego kroku $p.4$. W algorytmie LS2S, w odpowiadającym mu kroku $p.2$, wykorzystano ortogonalizację Grama-Schmidta. W przypadku SLS2S, otrzymane za pomocą tej metody rozwiązanie niestety nie będzie spełniało dodatkowych, narzuconych warunków dotyczących poszukiwanego wektora. Należy więc odpowiedzieć na pytanie: jaki zastosować algorytm umożliwiający wyznaczenie poszukiwanego wektora, spełniającego wszystkie warunki włącznie z maksymalizacją szerokości prześwitu ε, czyli składowej $(n+1)$ tego wektora?

Dodatkowo, ze względu na fakt, że krok $p.4$ jest najbardziej obciążającym obliczeniowo krokiem algortymu SLS2S, warto zastanowić się również nad tym, czy istnieje, mało kosztowna obliczeniowo, metoda przyśpieszenia działania algorytmu (w sensie liczby wykonań kroku $p.4$)? W rozdziale tym zostaną poruszone wszystkie te kwestie.

4.1. Element optymalizacji liczby wykonań kroku $p.4$ algorytmu SLS2S

Prostą metodą zmniejszenia liczby kroków niezbędnych do znalezienia rozwiązania zapewniającego ścisłą rozdzielność zbiorów jest umieszczenie na dwóch pierwszych miejscach listy Y_0 punktów, które stanowią parę, z całego zbioru X, najbliżej położonych względem siebie punktów z różnych klas.

Wynika to z faktu, że nie jest możliwe znalezienie hiperpłaszczyzn h_1 i h_2, bardziej odległych od siebie, niż wyznaczona minimalna odległość między wspomnianymi punktami.

Poniżej przedstawiono wpływ tego prostego zabiegu dla przykładu pierwszego, zaprezentowanego w części 3.2.3.. Algorytm SLS2S wykonano dla następującego uporządkowania:

$$Y_0 = \{f^*(\boldsymbol{x}_4), f^*(\boldsymbol{x}_6), f^*(\boldsymbol{x}_5), f^*(\boldsymbol{x}_2), f^*(\boldsymbol{x}_3), f^*(\boldsymbol{x}_1), f^*(\boldsymbol{x}_7), f^*(\boldsymbol{x}_8)\}$$

SLS2S(Y_0, $\emptyset$, $\emptyset$):

0.1 $K = \emptyset$

0.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

0.3 warunek spełniony $K = K \cup \{\boldsymbol{y}_6\} = \{\boldsymbol{y}_4, \boldsymbol{y}_6\}$

0.4
$$\left\{\begin{array}{rcl}\langle \boldsymbol{a}^*, \boldsymbol{y}_4\rangle & = & 0\\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6\rangle & = & 0\\ \alpha_1^2+\alpha_2^2 & = & 1\\ max(\varepsilon) & & \end{array}\right. \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{10}}{10}\begin{bmatrix}-3\\-1\\10\\7\end{bmatrix}$$

0.5 $ZLE_0 = \{\boldsymbol{y}_5\}$

0.6 warunek niespełniony

0.7 warunek niespełniony

0.8 warunek niespełniony

0.9 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_6, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7\boldsymbol{y}_8\}$

0.10 $\boldsymbol{b} = \boldsymbol{y}_5$

0.11 $c = 2$;

0.12 warunek spełniony, $K_1^{(1)} = K_0^{(1)} = \emptyset$, $K_1^{(2)} = K_0^{(2)} \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_5\}$

0.13 warunek niespełniony

0.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_1, K_1^{(1)}, K_1^{(2)})$

1.1 $K = \emptyset$

1.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

1.3 warunek niespełniony

1.4
$$\left\{\begin{array}{rcl}\langle \boldsymbol{a}^*, \boldsymbol{y}_4\rangle & = & 0\\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5\rangle & = & 0\\ \alpha_1^2+\alpha_2^2 & = & 1\\ max(\varepsilon) & & \end{array}\right. \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{17}}{17}\begin{bmatrix}-1\\-4\\17\\6\end{bmatrix}$$

1.5 $ZLE_1 = \{\boldsymbol{y}_6, \boldsymbol{y}_2, \boldsymbol{y}_7, \boldsymbol{y}_8\}$

1.6 warunek niespełniony

1.7 warunek niespełniony

1.8 warunek niespełniony

1.9 $Y_2 = \{\boldsymbol{y}_4, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

1.10 $\boldsymbol{b} = \boldsymbol{y}_6$

1.11 $c = 2$;

1.12 warunek spełniony, $K_2^{(1)} = K_1^{(1)} = \emptyset$, $K_2^{(2)} = K_1^{(2)} \cup \{\boldsymbol{y}_6\} = \{\boldsymbol{y}_5, \boldsymbol{y}_6\}$

1.13 warunek niespełniony

1.14 wykonaj algorytm $(\boldsymbol{a}^*, flag) \leftarrow \text{SLS2S}(Y_2, K_2^{(1)}, K_2^{(2)})$

2.1 $K = \emptyset$

2.2 warunek spełniony $K = \{\boldsymbol{y}_4\}$

2.3 warunek niespełniony

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5 \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6 \rangle &= 0 \\ \alpha_1^2 + \alpha_2^2 &= 1 \\ max(\varepsilon) \end{cases} \Rightarrow \quad \boldsymbol{a}^* = \frac{\sqrt{13}}{13} \begin{bmatrix} -3 \\ -2 \\ 11 \\ 8 \end{bmatrix}$$

2.5 $ZLE_2 = \emptyset$

2.6 warunek niespełniony

2.7 warunek spełniony

2.18 warunek niespełniony

1.15 warunek niespełniony

1.16 $ZLE_1 = \emptyset$

1.17 warunek niespełniony

1.18 warunek niespełniony

0.15 warunek niespełniony

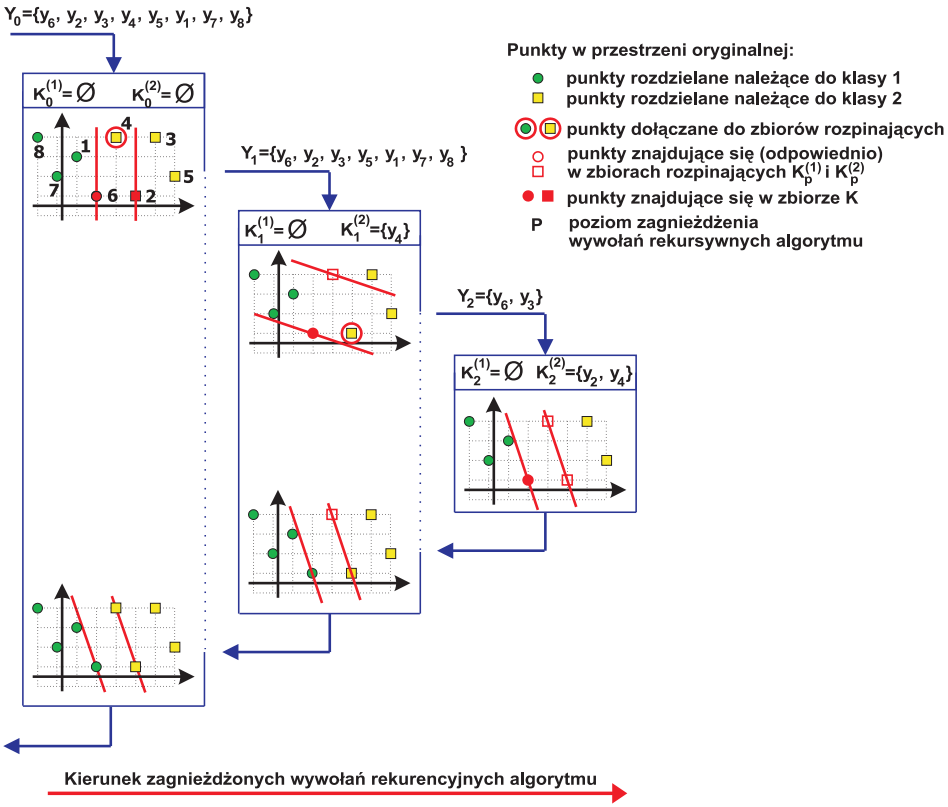
0.16 $ZLE_0 = \emptyset$

0.17 warunek niespełniony

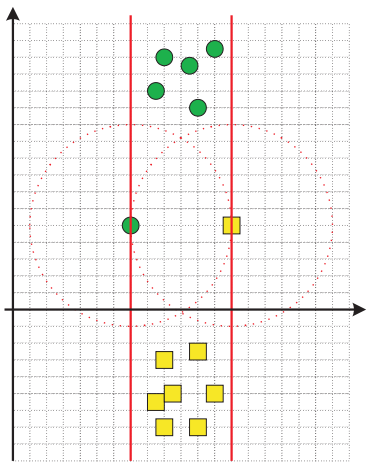
0.18 warunek spełniony — $\boldsymbol{a}^*$ jest poszukiwanym rozwiązaniem.

Analogicznie, w celu porównania, na rysunku 4.1 przedstawiono schemat wykonania algorytmu SLS2S z omówioną modyfikacją, dla zbioru z rysunku 3.3.

Choć w większości przypadków, ta mało kosztowna modyfikacja powoduje znaczące przyspieszenie działania algorytmu, nie jest to jednak reguła. Na rysunku 4.2 przedstawiono przykład zbiorów, gdzie wprowadzenie tej modyfikacji nie spowoduje wspomnianego efektu w tak dużym stopniu, jak w umieszczonych powyżej przykładach. Tym nie mniej należy podkreślić, że wprowadzona modyfikacja w żadnym z przypadków nie powoduje wydłużenia działania algorytmu (w sensie liczby wykonań kroku $p.4$).



Rysunek 4.1: Przykładowy schemat działania algorytmu SLS2S



Rysunek 4.2: Przykład zbiorów do rozdzielenia zmodyfikowanym algorytmem SLS2S

4.2. Implementacja kroku $p.4$ algorytmu SLS2S

4.2.1. Implementacja kroku $p.4$ algorytmu SLS2S dla $p=0, 1, 2$

W tym podpunkcie zaproponujemy sposób wyliczenia wektora $\boldsymbol{a}^*$ w kroku $p.4$ dla:

- $p=0,1$ oraz
- $p=2$ przy założeniu, że $\mathrm{card}(K_2^{(1)})=\mathrm{card}(K_2^{(2)})=1$.

Wymienione przypadki oznaczają, że lista rozpinająca hiperpłaszczyzny rozdzielające jest listą minimalną i spełniony jest warunek:

$$\mathrm{card}(K \cup K_p^{(1)} \cup K_p^{(2)}) = 2$$

W tych przypadkach po wykonaniu kroków $p.1$, $p.2$ i $p.3$ algorytmu, symbolicznie i bez utraty ogólności, możemy zapisać zawartość listy rozpinającej, jako:

$$K \cup K_p^{(1)} \cup K_p^{(2)} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}\}$$

gdzie $class(\boldsymbol{y}^{(1)})=1$ i $class(\boldsymbol{y}^{(2)})=2$. Zakładamy również, że $f^{*-1}(\boldsymbol{y}^{(1)}) \neq f^{*-1}(\boldsymbol{y}^{(2)})$.

W kroku $p.4$ dla wymienionych przypadków poszukiwany wektor $\boldsymbol{a}^*$ powinien spełniać następujące warunki:

$$\begin{cases} \left\langle \boldsymbol{a}^*, \boldsymbol{y}^{(1)} \right\rangle &= 0 \\ \left\langle \boldsymbol{a}^*, \boldsymbol{y}^{(2)} \right\rangle &= 0 \\ \sum_{i=1}^{n} (a_i^*)^2 &= 1 \\ max(\varepsilon) \end{cases} \tag{4.1}$$

Jeśli przyjmiemy na potrzeby tych rozważań, że $\boldsymbol{x}^{(1)} = f^{*-1}(\boldsymbol{y}^{(1)})$ i $\boldsymbol{x}^{(2)} = f^{*-1}(\boldsymbol{y}^{(2)})$, to zgodnie z intuicją (podrozdziały 3.2.1., 4.1.), potwierdzoną dowodem (dodatek A), wektor normalny hiperpłaszczyzn rozdzielających to wektor $\overrightarrow{\boldsymbol{x}^{(2)}\boldsymbol{x}^{(1)}}$. Długość zaś tego wektora określa wartość prześwitu ε.

Oznacza to, że przyjmując następujące oznaczenia:

$$\begin{aligned} \boldsymbol{y}^{(1)} &= \left[\ \boldsymbol{v}_1^{(1)},\ \ldots,\ \boldsymbol{v}_n^{(1)},\ 0,\ 1\ \right] = \left[\ \boxed{\boldsymbol{v}^{(1)}},\ 0,\ 1\ \right] \\ \boldsymbol{y}^{(2)} &= \left[\ -\boldsymbol{v}_1^{(2)},\ \ldots,\ -\boldsymbol{v}_n^{(2)},\ -1,\ -1\ \right] = \left[\ \boxed{-\boldsymbol{v}^{(2)}},\ -1,\ -1\ \right] \\ \boldsymbol{v} &= \boldsymbol{v}^{(1)} - \boldsymbol{v}^{(2)} \end{aligned}$$

poszukiwany wektor $\boldsymbol{a}^*$ jest równy:

$$\boldsymbol{a}^* = \frac{1}{\sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle}} \left[\ \boxed{\boldsymbol{v}},\ \langle \boldsymbol{v}, \boldsymbol{v} \rangle,\ -\left\langle \boldsymbol{v}, \boldsymbol{v}^{(1)} \right\rangle \right] \tag{4.2}$$

gdzie $\langle ., . \rangle$ oznacza iloczyn skalarny.

W implementacji maszynowej algorytmu SLS2S, w omawianych przypadkach, wystarczy przyjąć jako poszukiwany wektor, który opisuje jednoznacznie hiperpłaszczyzny rozdzielające, nieznormalizowany wektor:

$$\boldsymbol{a}^{*\prime} = \left[\ \boxed{\boldsymbol{v}},\ \langle \boldsymbol{v}, \boldsymbol{v} \rangle,\ -\left\langle \boldsymbol{v}, \boldsymbol{v}^{(1)} \right\rangle \right] \tag{4.3}$$

4.2.2. Implementacja kroku $p.4$ algorytmu SLS2S dla $p \geqslant 2$

W tym podpunkcie zaproponujemy sposób wyliczenia wektora $\boldsymbol{a}^*$ w kroku $p.4$ dla:

- $2 < p < n+1$ oraz
- $p = 2$ przy założeniu, że albo $\text{card}(K_2^{(1)}) = 2$ albo $\text{card}(K_2^{(2)}) = 2$.

W tych przypadkach po wykonaniu kroków $p.1$, $p.2$ i $p.3$ rozmiar listy rozpinającej jest następujący:

$$\text{card}\left(K \cup K_p^{(1)} \cup K_p^{(2)}\right) = p_1 + p_2$$

gdzie:

- p_1 — oznacza liczbę punktów, dla których $class(\boldsymbol{y}_i) = 1$ $(i = 1, \ldots, p_1)$,
- p_2 — liczba punktów, dla których $class(\boldsymbol{y}_i) = 2$ $(i = p_1+1, \ldots, (p_1+p_2))$.

Oznacza to, że wyznaczany w kroku $p.4$ wektor $\boldsymbol{a}^*$ ma spełniać następujący układ równań i warunek:

$$\begin{cases} \begin{bmatrix} y_{1,1} & \cdots & y_{1,p_1+p_2} & \cdots & y_{1,n} & 0 & 1 \\ y_{2,1} & \cdots & y_{2,p_1+p_2} & \cdots & y_{2,n} & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{p_1,1} & \cdots & y_{p_1,p_1+p_2} & \cdots & y_{p_1,n} & 0 & 1 \\ y_{p_1+1,1} & \cdots & y_{p_1+1,p_1+p_2} & \cdots & y_{p_1+1,n} & -1 & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{p_1+p_2,1} & \cdots & y_{p_1+p_2,p_1+p_2} & \cdots & y_{p_1+p_2,n} & -1 & -1 \end{bmatrix} \begin{bmatrix} a_1^* \\ a_2^* \\ \vdots \\ a_n^* \\ \varepsilon \\ a_{n+2}^* \end{bmatrix} = 0 \\ \sum_{i=1}^{n} (a_i^*)^2 = 1 \\ \max(\varepsilon) \end{cases} \tag{4.4}$$

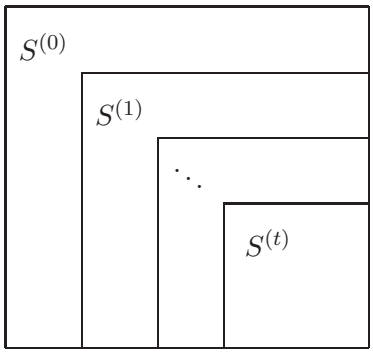
W implementacji maszynowej dla wyznaczania wektora $\boldsymbol{a}^*$ wykorzystano dwa algorytmiczne elementy: eliminiację Gaussa [36] oraz informacje zawarte w dowodzie poprawności algorytmu (dodatek A). I tak, w odniesieniu do równania macierzowego układu (4.4), należy zastosować zmodyfikowaną metodę eliminacji Gaussa. Modyfikacja polega na tym, że biorąc pod uwagę strukturę dwóch ostatnich kolumn macierzy po lewej stronie równania, jako pierwszy należy wyeliminować czynnik a_{n+2}^*, ponieważ nie występuje on w pozostałych elementach układu; dodatkowo jest to prosty zabieg powodujący wyrugowanie tego czynnika z pozostałych równań układu.

Jeżeli po zastosowaniu eliminacji Gaussa możemy stwierdzić, że $\varepsilon = 0$, to oznacza to, że otrzymane podczas eliminacji Gaussa wiersze macierzy pozwalają wyznaczyć p_1+p_2-2 pierwszych czynników a_i^* w zależności od pozostałych $n-(p_1+p_2-2)$ czynników. Ze względu na to, że $\varepsilon = 0$, możemy w implementacji maszynowej w tym przypadku przyjąć $a_i^* = 1$ dla $i = (p_1+p_2-1)$, (p_1+p_2), $\ldots$, n; co pozwala na sprawne wyliczenie zależnych od tych czynników: a_i^* dla $i = 1, \ldots,$ (p_1+p_2-2).

W przypadku, gdy po zastosowaniu eliminacji Gaussa zachodzi nierówność $\varepsilon \neq 0$, wówczas należy wykorzystać informacje zawarte w dowodzie poprawności algorytmu (zależności A.13--A.16) i wyznaczyć macierz $\boldsymbol{S}^{(0)}$ (jest to macierz o maksymalnych wymiarach $n \times n$, minimalnie ma ona wymiary 2×2). Macierz ta określa współczynniki równania kwadratowego z układu (4.4) przedstawionego w postaci formy kwadratowej (A.15).

Macierz $\boldsymbol{S}^{(0)}$ jest podstawą do wyliczenia pozostałych macierzy $\boldsymbol{S}^{(i)}$ dla $i=1, 2, ..., t$, gdzie $t=n-(p_1+p_2)+1$. W implementacji tego etapu, uwzględniając fakt, że macierze te są macierzami symetrycznymi, możemy zminimalizować liczbę operacji matematycznych, wyznaczając elementy położone nad przekątną.

Uwzględniając własności macierzy $\boldsymbol{S}^{(i)}$ dla $i=0, 1, 2, ..., t$, wykazane w dowodzie wykonalności kroku $p.4$ (1.2.), w wyznaczeniu wektora $\boldsymbol{a}^*$, niezbędne jest przechowywanie tylko pierwszego wiersza każdej z macierzy $\boldsymbol{S}^{(i)}$ dla $i=0, 1, 2, ..., (t-1)$. Oznacza to, że macierze $\boldsymbol{S}^{(i)}$ dla $i=1, 2, ..., t$ można przechowywać wewnątrz macierzy $\boldsymbol{S}^{(0)}$ — ilustruje to poniższy schemat:



(4.5)

Ostatnia z wyznaczanych macierzy $\boldsymbol{S}^{(t)}$, gdzie $t=(n-(p_1+p_2)+1)$, ma następującą strukturę:

$$S^{(t)} = \begin{bmatrix} s_{1,1}^{(t)} & 0 \\ 0 & s_{2,2}^{(t)} \end{bmatrix} \tag{4.6}$$

Wówczas możliwe jest wyznaczenie wartości prześwitu:

$$\varepsilon = \sqrt{-\frac{s_{2,2}^{(t)}}{s_{1,1}^{(t)}}}$$

lub przyjęcie na potrzeby implementacji maszynowej wartości:

$$a_{n+1}^* = \varepsilon = -\frac{s_{2,2}^{(t)}}{s_{1,1}^{(t)}} \tag{4.7}$$

unikając tym samym pierwiastkowania. Otrzymane rozwiązanie jest wektorem spełniających wyjściowy układ równań i warunku przeskalowanym o wartość $\sqrt{-\frac{s_{1,1}^{(t)}}{s_{2,2}^{(t)}}}$.

Przyjmując następującą strukturę każdej z macierzy $\boldsymbol{S}^{(i)}$ dla $i=0, 1, \ldots, t$:

$$\boldsymbol{S}^{(i)} = \begin{bmatrix} s_{1,1}^{(i)} & \boldsymbol{b}^{(i)*\mathrm{T}} \\ \boldsymbol{b}^{(i)*} & \boldsymbol{c}^{(i)} \end{bmatrix} \tag{4.8}$$

i uwzględniając prawdziwość lematu 3 z równością (A.23) oraz wynikającego z niego faktu, że:

$$s_{1,1}^{(k)} = \Psi_k \det\left[\boldsymbol{C}_{k+1}\right],$$

gdzie macierz $\boldsymbol{C}_{k+1}$ jest zgodnie z definicją 7 (dodatek A), podmacierzą $\boldsymbol{S}^{(0)}$ uzyskaną przez skreślenie z niej wszystkich wierszy i kolumn, których indeksy są większe od wartości $(k+1)$, zaś współczynnik Ψ_k:

$$\Psi_k = -4\Psi_{k-1}^2 \det\left[\boldsymbol{C}_{k-1}\right], \text{ przy } \Psi_1 = -4,$$

uzyskamy stabilizację numeryczną przy wyznaczaniu kolejnych macierzy $\boldsymbol{S}^{(i)}$ dla $i = 1, 2, ..., t$, stosując następujące przekształcenie:

$$\boldsymbol{S}^{(i)} = \frac{\boldsymbol{b}^{(i-1)^*}\boldsymbol{b}^{(i-1)^{*T}} - s_{1,1}^{i-1}\boldsymbol{c}^{(i-1)}}{|\Psi'(i)|}, \qquad \text{gdzie} \quad \Psi'(i) = \begin{cases} 1 & i = 1 \\ s_{1,1}^{(i-2)} & i \neq 1 \end{cases} \tag{4.9}$$

Po wyznaczeniu macierzy $\boldsymbol{S}^{(i)}$ $(i = 1, \ldots, t)$ i wartości prześwitu ε, przystępujemy do wyznaczenia czynników: $a^*_{p_1+p_2+i}$ dla $i = (t-1), \ldots, 0$, stosując następujący wzór:

$$a^*_{p_1+p_2+i} = -\frac{\left\langle \boldsymbol{b}^{(i)^*}, \boldsymbol{v}^{i+1} \right\rangle}{s_{1,1}^{(i)}} \tag{4.10}$$

gdzie:

$$\boldsymbol{v}^{(i)} = \left[a^*_{p_1+p_2+i},\, a^*_{p_1+p_2+i+1},\, \ldots,\, a^*_n, \varepsilon,\, 1\right] \tag{4.11}$$

Wyznaczone w ten sposób czynniki, należy podstawić do zależności opisujących czynniki a_i^* dla $i = (p_1+p_2-1), \ldots, 2, 1$ oraz a^*_{n+2}, a które otrzymano po wykonaniu eliminacji Gaussa i wyznaczyć brakujące wartości.

Przedstawiony sposób wyznaczania wektora $\boldsymbol{a}^*$ w kroku $p.4$ wymaga znaczących zasobów pamięciowych. W etapie pierwszym wykorzystywana jest macierz o wymiarach $(p_1+p_2) \times (n+2)$. Wykorzystanie własności rozwiązywanego równania kwadratowego w układzie (4.4) umożliwia zminimalizowanie zużycia pamięci w etapie drugim, gdzie wykorzystana macierz ma wymiary $(n-(p_1+p_2)+3) \times (n-(p_1+p_2)+3)$.

Pojawia się więc pytanie: Jak można wykorzystać algorytm rozdzielający dwa zbiory w sposób optymalny, mając na uwadze jego, przedstawione do tej pory, własności?

Rozdział 5

Konstrukcja klasyfikatora dwu-decyzyjnego — dwa podejścia

Jednym ze sposobów konstruowania klasyfikatorów odcinkowo-liniowych jest wykorzystanie dwóch algorytmów:

- algorytmu rozdzielającego w sposób optymalny, jeśli zbiory są liniowo rozdzielne oraz
- algorytmu jednoznacznie określającego, co robić w przypadku, gdy zbiory nie są liniowo rozdzielne.

Algorytmy rozdzielające w sposób optymalny mogą stanowić inspirację do tworzenia drugiego z wymienionych algorytmów, jeżeli zadanie przez nie realizowane określimy w następujący sposób: znaleźć tzw. „nakładkę" dla zbiorów nieseparowalnych liniowo (rysunek 5.1).

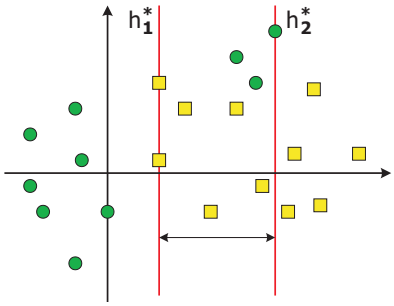
Definicja 6 *Dwa zbiory nierozdzielne liniowo są rozdzielne liniowo z nakładką ε, gdy istnieją funkcje $h_1^*(\boldsymbol{x})$ i $h_2^*(\boldsymbol{x})$ takie, że:*

$$\begin{cases} h_1^*(\boldsymbol{x}_1) \geqslant 0 \\ h_2^*(\boldsymbol{x}_2) \leqslant 0 \end{cases} \quad \boldsymbol{x}_1 \in X_1, \;\; \boldsymbol{x}_2 \in X_2$$

$$h_1^*(\boldsymbol{x}) = \sum_{i=1}^{n} a_i^* x_i + a_{n+2}^* \qquad h_2^*(\boldsymbol{x}) = \sum_{i=1}^{n} a_i^* x_i - \varepsilon + a_{n+2}^* \qquad \sum_{i=1}^{n} (a_i^*)^2 = 1, \quad \varepsilon > 0 \tag{5.1}$$

Obszar wyznaczony przez nierówności: $h_1^*(\boldsymbol{x}) > 0$ i $h_2^*(\boldsymbol{x}) < 0$ dla każdego $x \in E^n$ nazywać będziemy nakładką, wartość ε zaś szerokością nakładki.

Wprowadzenie rozdzielności z nakładką zapewnia, że wszystkie punkty ze zbioru uczącego, znajdujące się poza obszarem wyznaczonym przez nakładkę, byłyby poprawnie rozpoznane przez klasyfikator wykorzystujący wspomniane funkcje h_1^* i h_2^*. Algorytm wyznaczający nakładkę dla zbiorów nierozdzielnych liniowo wydaje się być dualnym do omawianego wcześniej algorytmu rozdzielającego zbiory z prześwitem ε. Analogicznie należy wyznaczyć parę hiperpłaszczyzn równoległych oddalonych od siebie o wartość ε, która w tym przypadku określa szerokość tej nakładki.



Rysunek 5.1: Zbiory rozdzielne z nakładką ε

Kolejnym krokiem w planowanych pracach miała być modyfikacja algorytmu wyznaczającego nakładkę tak, aby była minimalna. Prawidłowa definicja algorytmów rozdzielających w sposób optymalny i rozdzielających z minimalną nakładką, pozwalałaby na tworzenie hierarchicznych klasyfikatorów dla zbiorów o dowolnej strukturze. Algorytm tworzenia takiego klasyfikatora dla punktów ze zbioru $X = X_1 \cup X_2$ można by wówczas zapisać w postaci następującej listy kroków do wykonania:

Algorytm 1 ***dla parametru X zwraca listę L***

1. *X_a=X to zbiór aktualnie rozdzielanych punktów.*
2. *L to pusta lista, która będzie zawierała hierarchię hiperpłaszczyzn rozdzielających, opisujących jednoznacznie tworzony klasyfikator odcinkowo-liniowy.*
3. ***Wykonaj algorytm SLS2S** dla zbioru X_a.*
4. *Jeśli w wyniku wykonania algorytmu SLS2S okazało się, że punkty są rozdzielne liniowo, wówczas, otrzymane jako rozwiązanie, hiperpłaszczyzny rozdzielające dodaj do zbioru L. Idź do kroku 9.*
5. *Jeśli w wyniku wykonania algorytmu SLS2S okazało się, że punkty nie są rozdzielne liniowo, wówczas **wykonaj algorytm rozdzielania z nakładką z parametrem X_a**.*
6. *Otrzymane rozwiązanie dołącz na koniec listy L.*
7. *Usuń ze zbioru X_a wszystkie te punkty, które leżą poza obszarem nakładki.*
8. *Idź do kroku 3.*
9. *Na podstawie zbiorów X_1 i X_2 wyznacz przynależność do danej klasy obszarom utworzonym przez hierarchię hiperpłaszczyzn L.*

Otrzymana jako wynik lista L zawiera w każdym (poza ostatnim) elemencie, parę hiperpłaszczyzn dzielących przestrzeń na trzy obszary: dwa obszary, których punkty zostałyby zaklasyfikowane odpowiednio do jednej z dwóch klas oraz obszar, który jest dalej dzielony przez parę hiperpłaszczyzn zawartych w kolejnym węźle listy.

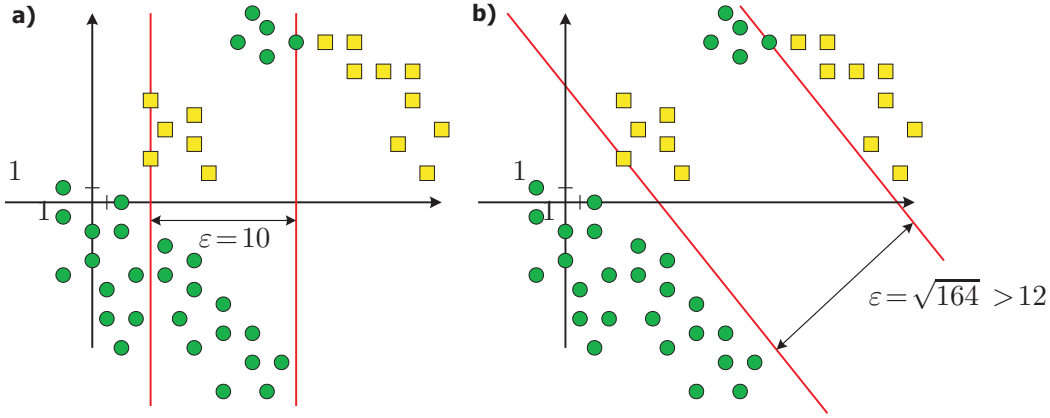
Ze względu na to, że kolejność tych par hiperpłaszczyzn jest istotna, w dalszej części pracy struktura przechowująca hiperpłaszczyzny rozdzielające będzie nazywana *hierarchią hiperpłaszczyzn rozdzielających*. W przypadku algorytmu 1 jest to struktura reprezentowana przez listę.

Pozostaje odpowiedzieć na pytanie: dlaczego przedstawione powyżej analogie pomiędzy algorytmem SLS2S a algorytmem rozdzielającym z minimalną nakładką, okazały się tylko pozornymi analogiami? To w konsekwencji uniemożliwiło implementację przedstawionego algorytmu 1.

5.1. Różne definicje minimalnej nakładki

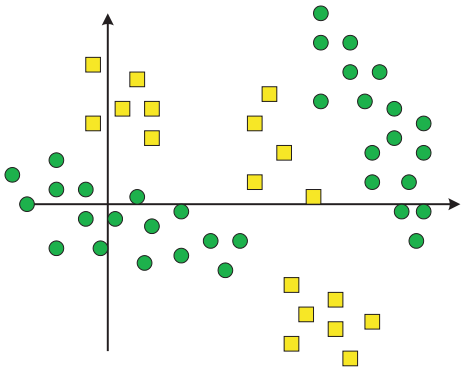
Jeśli przyjmiemy, że dobry klasyfikator odcinkowo-liniowy to taki, który ma w swej hierarchii minimalną liczbę hiperpłaszczyzn rozdzielających z nakładką (tzn. lista L z algorytmu 1 ma jak najmniejszą liczbę węzłów), to można zadać na nowo pytanie: jakie własności powinna mieć wówczas minimalna nakładka? Można przykładowo przyjąć, że zbiory rozdzielone z minimalną nakładką to takie, dla których nakładka:

- jest najwęższa (rysunek 5.2.a) lub
- zapewnia, że jak największa liczba punktów należących do zbioru uczącego może być poprawnie sklasyfikowana (rysunek 5.2.b).



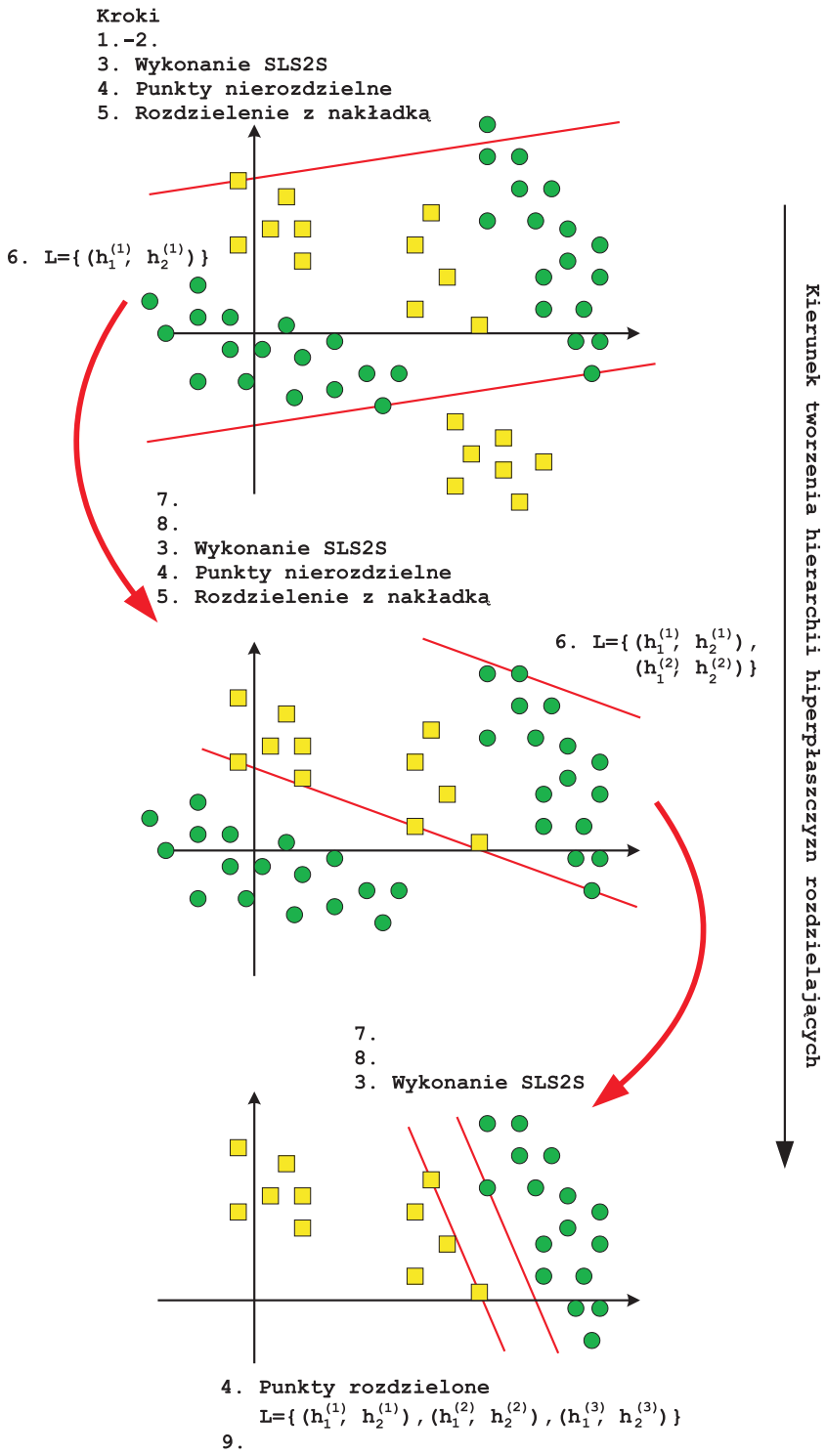
Rysunek 5.2: Zbiory rozdzielne z różnie określoną „minimalną” nakładką ε

Próba określenia własności nakładki rodzi kolejne pytanie: czy jest prawdą, że klasyfikator zbudowany z wykorzystaniem takiego algorytmu będzie klasyfikatorem, który w swej hierarchii będzie zawierał jak najmniejszą liczbę hiperpłaszczyzn rozdzielających? Podanie odpowiedzi nie jest łatwe, gdy dla przedstawionego na rysunku 5.3 przykładu próbujemy „ręcznie” określić, „najlepsze” według nas, hiperpłaszczyzny rozdzielające i opisać ich własności.



Rysunek 5.3: Przykład zbioru uczącego, dla którego należy zbudować klasyfikator

Na rysunku 5.4 przedstawiono schemat tworzenia hierarchii hiperpłaszczyzn rozdzielających z wykorzystaniem algorytmu 1. W algorytmie przyjęto, że nakładka ma być o jak najmniejszej szerokości.

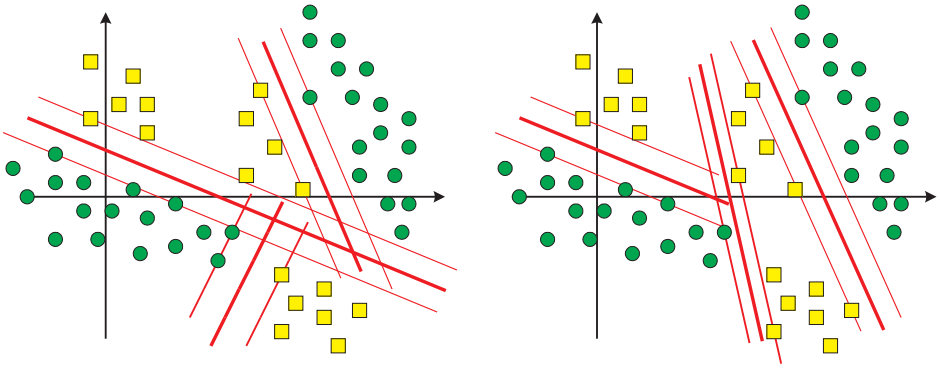


Rysunek 5.4: Schemat działania algorytmu 1 dla zbioru uczącego z rysunku 5.3

Przykład ten pokazuje dodatkowo, że sprawdzenie poprawności otrzymanego rozwiązania jest równie trudnym zadaniem, co podanie rozwiązania.

Nie jest też łatwe odpowiedzenie na pytanie, jaka własność nakładki byłaby wystarczająca, by otrzymać, po wykonaniu algorytmu 1, podane na rysunku 5.5 rozwiązania? Tymczasem właśnie te rozwiązania wydają się najbardziej „naturalne”. Równocześnie łatwo jest sprawdzić,

że są to klasyfikatory o minimalnej liczbie hiperpłaszczyzn rozdzielających bez uwzględniania ich hierarchii. Dodatkowo są to rozwiązania, których uzyskanie wydaje się prostsze poprzez zastosowanie opisywanego algorytmu SLS2S niż wspomnianych algorytmów z nakładką.

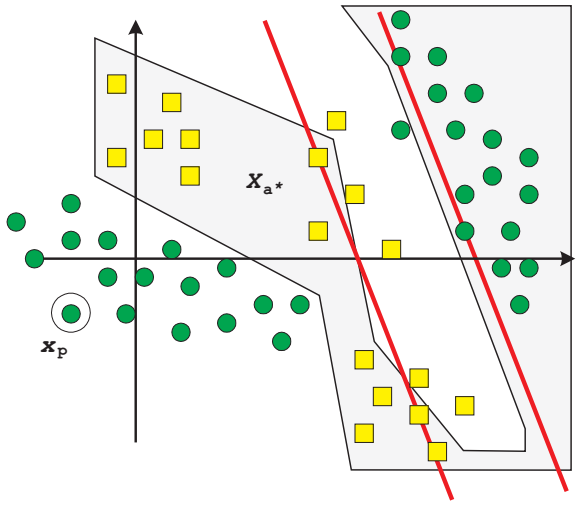


Rysunek 5.5: Przykłady hiperpłaszczyzn rozdzielających klasyfikatorów odcinkowo-liniowych

5.2. Konstrukcja klasyfikatora odcinkowo-liniowego, wykorzystującego algorytm SLS2S

5.2.1. Idea konstrukcji klasyfikatora

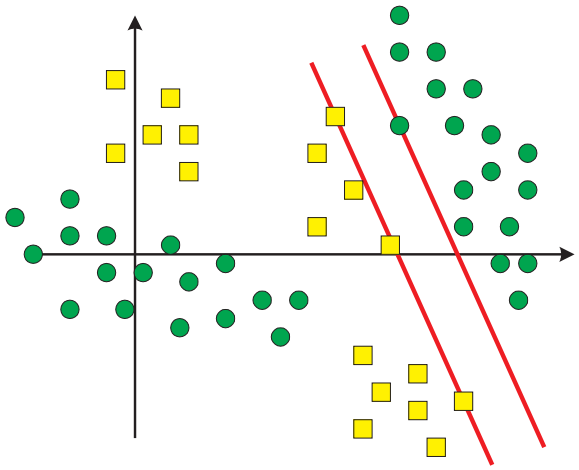
W przypadku użycia algorytmu SLS2S, gdy rozdzielane zbiory nie są separowalne liniowo, dysponować możemy następującą wiedzą. Znany jest ostatni wyznaczony wektor $\boldsymbol{a}^*$ (opisujący jednoznacznie hiperpłaszczyzny rozdzielające h_1 i h_2), który poprawnie rozdziela część punktów $X_{a^*} \subset X = X_1 \cup X_2$ oraz punkt $\boldsymbol{x}_p$, którego dołączenie do grupy dobrze rozdzielanych punktów nie było możliwe. Sytuację tą ilustruje rysunek 5.6.



Rysunek 5.6: Ostatni etap rozdzielania zbiorów nierozdzielnych liniowo za pomocą SLS2S

Wszystkie nieprawidłowo rozdzielone na tym etapie punkty możemy podzielić na dwie grupy. Jedną, stanowią te punkty, które z powodzeniem moglibyśmy dołączyć do zbioru prawidłowo rozdzielanych punktów. Druga, to grupa punktów, które tak jak punkt $\boldsymbol{x}_p$ uniemożliwiają liniowe rozdzielenie zbiorów. Zamiast kończyć działanie algorytmu SLS2S rozdzielmy zbiory na tyle,

na ile jest to możliwe, dołączając punkty z wymienionej pierwszej grupy. Otrzymane w ten sposób rozwiązanie, choć nie zapewnia rozdzielności liniowej, ponieważ nie jest to możliwe, powoduje, że otrzymane hiperpłaszczyzny dzielą przestrzeń na trzy części o następujących własnościach (rysunek 5.7). Po pierwsze, między hiperpłaszczyznami rozdzielającymi nie znajduje się żaden punkt. Gdyby był, byłoby możliwe jego dołączenie. Po drugie, w najgorszym przypadku w pozostałych dwóch częściach znajdują się rozdzielne podzbiory zbioru X, które zgodnie z zasadą *dziel i zwyciężaj* [23] możemy rozdzielić wykonując przynajmniej dwa razy algorytm SLS2S.



Rysunek 5.7: Własności hiperpłaszczyzn rozdzielających zbiory nierozdzielne

Podkreślmy, że wymienione własności nie zależą od charakteru rozdzielanych zbiorów (np. liczby i sposobu rozłożenia klastrów), co stanowi ogromną zaletę. Należy bowiem zwrócić uwagę, że choć własności te są łatwe do określenia w przypadku zbiorów dwuwymiarowych, w praktycznych zastosowaniach, gdzie wymiarowość danych jest dużo większa, stanowią zakres osobnych badań naukowych.

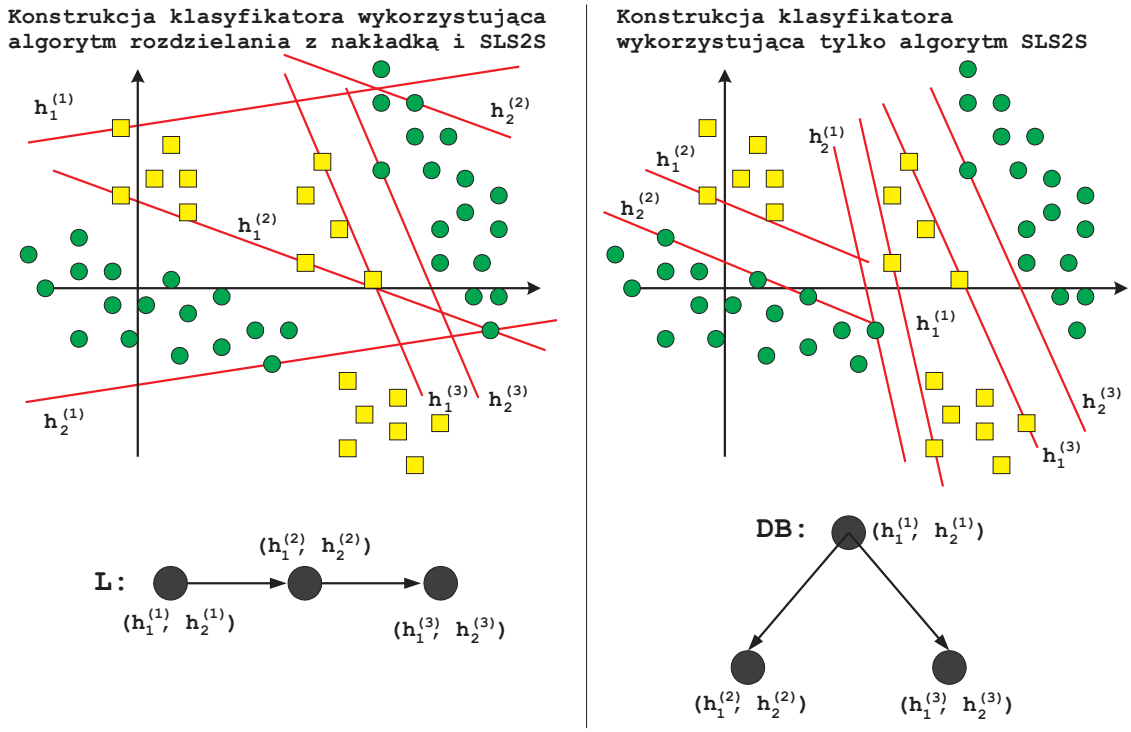
5.2.2. Struktura danych opisujących klasyfikator

Przedstawione techniki konstrukcji klasyfikatora odcinkowo-liniowego wymagają różnych struktur danych do przechowywania hierarchii hiperpłaszczyzn rozdzielających.

W przypadku klasyfikatora, do którego konstrukcji wykorzystano algorytm SLS2S i algorytm rozdzielania z nakładką, hierarchia hiperpłaszczyzn rozdzielających przechowywana jest w strukturze listowej — jednokierunkowej liście. Nie jest potrzebne przechowywanie informacji dotyczącej, który z dwóch algorytmów w danym węźle listy został zastosowany. Jeżeli lista zawiera k węzłów, to w $(k-1)$ pierwszych węzłach użyto algorytmu rozdzielania z nakładką, zaś w ostatnim węźle — algorytm SLS2S.

Jeśli do konstrukcji klasyfikatora wykorzystano tylko i wyłącznie algorytm SLS2S, to hierarchia hiperpłaszczyzn rozdzielających zawarta powinna być w strukturze drzewiastej — drzewie binarnym.

Przykładowe hierarchie przedstawiono na rysunku 5.8.



Rysunek 5.8: Struktury danych opisujące klasyfikator

5.2.3. Szkic algorytmu LC2S tworzącego klasyfikator

Wymienione w punkcie 5.2.1. własności pozwalają na określenie drugiego algorytmu tworzenia klasyfikatora odcinkowo-liniowego dla rozdzielanych zbiorów X_1 i X_2. Algorytm LC2S (*Linear Classifier of two Sets*) zwraca węzeł będący korzeniem drzewa hierarchii hiperpłaszczyzn rozdzielających.

Algorytm 2 ***[szkic algorytmu LC2S]***
dla parametrów X_1 i X_2 zwraca korzeń drzewa DB

1. *Utwórz węzeł DB drzewa binarnego hierarchii hiperpłaszczyzn rozdzielających zbiory X_1 i X_2.*
2. *Rozdziel tyle, ile jest możliwych punktów należących do zbiorów X_1 i X_2 przy pomocy algorytmu SLS2S. Otrzymany wektor $\boldsymbol{a}^*$ jednoznacznie opisuje hiperpłaszczyzny h_1 i h_2.*
3. *Zapisz wektor $\boldsymbol{a}^*$ w węźle DB.*
4. *$Z^{(1)}$ — zbiór źle rozdzielonych punktów należących do X_1.*
5. *$Z^{(2)}$ — zbiór źle rozdzielonych punktów należących do X_2.*
6. *Jeśli wszystkie punkty zostały prawidłowo rozdzielone tzn. $card(Z^{(1)}) = card(Z^{(2)}) = 0$, idź do punktu 9.*
7. *Jeśli $card(Z^{(1)}) \neq 0$, wówczas w węźle DB utwórz lewego potomka i umieść tam węzeł otrzymany po wykonaniu algorytmu **Algorytm 2 z parametrami $Z^{(1)}$ i $X_2 \backslash Z^{(2)}$**.*
8. *Jeśli $card(Z^{(2)}) \neq 0$, wówczas w węźle DB utwórz prawego potomka i umieść tam węzeł otrzymany po wykonaniu algorytmu **Algorytm 2 z parametrami $X_1 \backslash Z^{(1)}$ i $Z^{(2)}$**.*
9. *Zwróć jako wynik działania algorytmu węzeł DB.*

5.2.4. Algorytm LC2S

Przedstawiany algorytm LC2S podobnie jak algorytm SLS2S ma następujące parametry wejściowe:

- Y_p — lista rozdzielanych punktów,
- $K_p^{(1)}$ — lista punktów, przez które ma przechodzić hiperpłaszczyzna h_1,
- $K_p^{(2)}$ — lista punktów, przez które ma przechodzić hiperpłaszczyzna h_2.

i dwa parametry wyjściowe:

- $flag$ — wartość 1 oznacza, że zbiory są rozdzielne liniowo; wartość -1 oznacza, że zbiory nie są rozdzielne; wartość ma znaczenie dla tych rekurencyjnych wywołań algorytmu, gdzie $p>0$.
- DB — korzeń drzewa, reprezentujący hierarchię hiperpłaszczyzn rozdzielających, gdzie każdy z węzłów drzewa zawiera wektor $\boldsymbol{a}^*$ jednoznacznie określający hiperpłaszczyzny h_1 i h_2.

Na starcie algorytmu $p=0$. Wszystkie punkty, które należy rozdzielić tworzą listę Y_0:

$$Y_0=\{f^*(\boldsymbol{x})\in E^{n+2}\colon\ \boldsymbol{x}\in X_1\cup X_2\in E^n\}.$$

Nieznane są punkty rozpinające hiperpłaszczyzny rozdzielające, więc $K_0^{(1)}=\emptyset$ i $K_0^{(2)}=\emptyset$.

Parametr p w numeracji kroków algorytmu jest istotny. Z jednej strony, podobnie jak w algorytmie SLS2S zwiększa czytelność; z drugiej, co ważniejsze, pewne kroki algorytmu dla $p=0$ są odzwierciedleniem kroków szkicu algorytmu przedstawionego w punkcie 5.2.3.. Wszystkie pozostałe kroki, gdy $p\neq 0$, są realizacją kroku 2 wspomnianego szkicu.

Nagłówek algorytmu LC2S: $(DB, flag) \leftarrow \mathrm{LC2S}(Y_p, K_p^{(1)}, K_p^{(2)})$
$$p=\mathrm{card}\left(K_p^{(1)}\cup K_p^{(2)}\right)$$

Definicja algorytmu:

p.1 Utwórz węzeł $DB^{(p)}$ drzewa binarnego.

p.2 $flag=1$ — znacznik informujący, czy zbiory są rozdzielne liniowo.
$Z=\emptyset$ — lista punktów pomijanych przy rozdzielaniu — punkty, o których wiadomo już, że nie mogą być dołączone do zbioru punktów poprawnie rozdzielonych.

p.3 Jeśli $\mathrm{card}\left(K_p^{(1)}\right)=0$, wówczas $K=\{$pierwszy $\boldsymbol{y}\in Y_p$ dla którego $class(\boldsymbol{y})=1\}$
w przeciwnym razie $K=\emptyset$.

p.4 Jeśli $\mathrm{card}\left(K_p^{(2)}\right)=0$, wówczas $K=K\cup\{$pierwszy $\boldsymbol{y}\in Y_p$ dla którego $class(\boldsymbol{y})=2\}$.

p.5 $\boldsymbol{a}^*$ — wektor w przestrzeni E^{n+2} z maksymalną wartością składnika ε, prostopadły do każdego wektora zawartego na liście „rozpinającej”: $K\cup K_p^{(1)}\cup K_p^{(2)}$.

p.6 $ZLE_p=\{\boldsymbol{y}\in Y_p\backslash Z\colon \langle\boldsymbol{a}^*,\boldsymbol{y}\rangle<0\}$, tzn. aktualna lista źle rozdzielonych punktów.

p.7 Jeśli $\varepsilon=0$ i $\mathrm{card}(ZLE_p)>\frac{\mathrm{card}(Y_p)}{2}$ wówczas zmień kierunek wektora $\boldsymbol{a}^*$ na przeciwny $(\boldsymbol{a}^*\leftarrow-\boldsymbol{a}^*)$ i idź do kroku *p*.6.

p.8 Jeśli $p=n+1$ i $ZLE_p\neq\emptyset$, ustaw $flag=-1$ i idź do kroku *p*.22 — zbiory nie są rozdzielne liniowo.

p.9 Jeśli $flag = 1$ zapisz $\boldsymbol{a}^*$ w węźle $DB^{(p)}$, tzn. $DB^{(p)}_{\boldsymbol{a}^*} = \boldsymbol{a}^*$.

p.10 Jeśli $ZLE_p = \emptyset$, tzn. wszystkie „możliwe do rozdzielenia" punkty zostały poprawnie rozdzielone, to idź do kroku *p*.18, w przeciwnym przypadku wyznacz wektor $\boldsymbol{b}$ z listy ZLE_p w następujący sposób:

$$\boldsymbol{b} = \begin{cases} ||\boldsymbol{b}, \boldsymbol{b}_K|| = \min\limits_{1 \leqslant i \leqslant m} ||\boldsymbol{b}_i, \boldsymbol{b}_K|| & \text{gdy } p = 0 \\ \text{pierwszy wektor z listy } ZLE_p & \text{gdy } p \neq 0 \end{cases}$$

gdzie $m = \text{card}\,(ZLE_p)$, $\boldsymbol{b}_K \in K$ i $class(\boldsymbol{b}_i) = class(\boldsymbol{b}_K)$.

p.11 $c = class\,(\boldsymbol{b})$.

p.12 $Y_{p+1} = Y_p \setminus (ZLE_p \cup Z)$.

p.13 Jeśli $\text{card}\left(K_p^{(c)}\right) = n$, ustaw $flag = -1$ i idź do kroku *p*.22 (zbiory nie są rozdzielne liniowo), w przeciwnym razie dodaj do odpowiedniego fragmentu listy rozpinającej: $K_{p+1}^{(c)} = K_p^{(c)} \cup \{\boldsymbol{b}\}$, $K_{p+1}^{(3-c)} = K_p^{(3-c)}$.

p.14 Wywołaj rekurencyjnie algorytm LC2S: $(DB, flag) \leftarrow \text{LC2S}\ (Y_{p+1}, K_{p+1}^{(1)}, K_{p+1}^{(2)})$.

p.15 Jeśli $flag = 1$ pobierz przechowywany w zwróconym węźle *DB* wektor $\boldsymbol{a}^*$ i zapisz w węźle $DB^{(p)}$, tzn. $DB^{(p)}_{\boldsymbol{a}^*} = DB_{\boldsymbol{a}^*}$ i potraktuj jako aktualne rozwiązanie $\boldsymbol{a}^*$. Wyznacz ponownie źle rozdzielone punkty:

$$ZLE_p = \{\boldsymbol{y} \in Y_p \setminus Z : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$$

i idź do kroku *p*.10.

p.16 Jeśli $flag = -1$ i $p \neq 0$ idź do kroku *p*.22.

p.17 Dołącz $\boldsymbol{b}$ do zbioru punktów, które nie mogą być dodane do zbioru poprawnie rozdzielonych, tzn. $Z = Z \cup \{\boldsymbol{b}\}$. Usuń ten punkt ze zbioru źle rozdzielonych punktów $ZLE_p = ZLE_p \setminus \{\boldsymbol{b}\}$. Idź do kroku *p*.10.

p.18 Jeśli $p \neq 0$ lub $Z = \emptyset$, wówczas idź do kroku *p*.22.

p.19 $Y_0^{(1)} = \{\boldsymbol{y} \in Y_0 :\ class(\boldsymbol{y}) = 1\}$, $\quad Z^{(1)} = \{\boldsymbol{y} \in Z :\ class(\boldsymbol{y}) = 1\}$,
$Y_0^{(2)} = \{\boldsymbol{y} \in Y_0 :\ class(\boldsymbol{y}) = 2\}$, $\quad Z^{(2)} = \{\boldsymbol{y} \in Z :\ class(\boldsymbol{y}) = 2\}$.

p.20 Jeśli $\text{card}(Z^{(1)}) \neq 0$ utwórz lewego potomka $DB_L^{(p)}$ węzła $DB^{(p)}$ i przypisz mu węzeł zwrócony jako wynik wykonania algorytmu $\text{LC2S}(Z^{(1)} \cup \left(Y_0^{(2)} \setminus Z^{(2)}\right), \emptyset, \emptyset)$.

p.21 Jeśli $\text{card}(Z^{(2)}) \neq 0$ utwórz prawego potomka $DB_R^{(p)}$ węzła $DB^{(p)}$ i przypisz mu węzeł zwrócony jako wynik wykonania algorytmu $\text{LC2S}(Z^{(2)} \cup \left(Y_0^{(1)} \setminus Z^{(1)}\right), \emptyset, \emptyset)$.

p.22 Zwróć jako wynik węzeł $DB^{(p)}$ i znacznik $flag$.

Przedstawiony algorytm zasługuje na kilka zdań komentarza. Poniższa tabela zawiera zestawienie kroków algorytmu LC2S, realizujących bezpośrednio odpowiednie kroki przedstawionego algorytmu 2. Wszystkie nie wymienione w tabeli kroki algorytmu LC2S i krok *p*.18 dla $p = 0$ realizują algorytm SLS2S, a tym samym krok drugi algorytmu 2.

algorytm LC2S	komentarz	algorytm 2
$p.1$	Tworzenie węzła drzewa binarnego.	krok 1
$p.19$	Określenie zbiorów źle rozdzielonych punktów należących odpowiednio do X_1 i X_2.	kroki 4 i 5
$p.9$, $p.15$	Zapamiętanie aktualnego poprawnego rozwiązania.	krok 3
$p.18$	Dla $p=0$ jest to sprawdzenie, czy wszystkie punkty zostały prawidłowo rozdzielone.	krok 6
$p.20$	Utworzenie lewego potomka w drzewie binarnym i uruchomienie algorytmu z odpowiednimi parametrami.	krok 7
$p.21$	Utworzenie prawego potomka w drzewie binarnym i uruchomienie algorytmu z odpowiednimi parametrami.	krok 8
$p.22$	Zwrócenie rezultatu działania algorytmu.	krok 9

Złożoność pamięciowa algorytmu LC2S jest dokładnie taka sama jak złożoność algorytmu SLS2S, powiększona o pamięć potrzebną do opisu struktury drzewa binarnego.

Najistotniejszym czynnikiem pesymistycznej złożoności obliczeniowej tego algorytmu jest złożoność obliczeniowa algorytmu SLS2S. Oznacza to, że wszelkie próby optymalizacji algorytmu LC2S w rzeczywistości powinny dotyczyć optymalizacji algorytmu SLS2S.

Czy istnieją przesłanki sugerujące celowość prowadzenia dalszych badań w zakresie optymalizacji algorytmu SLS2S, a tym samym również algorytmu LC2S?

Podsumowanie

W kolejnych rozdziałach rozprawy opisano przeprowadzone do tej pory prace związane z określeniem i wykorzystaniem algorytmu badania optymalnej rozdzielności liniowej dwóch zbiorów. Istotnym elementem, na który kładziono nacisk, było zachowanie chronologii prowadzonych prac i pojawiających się „kłopotów". „Kłopoty" te bowiem zwykle były ukrytym źródłem dobrych „wiadomości". Biorąc pod uwagę tylko ten aspekt, praca ta może być ciekawym materiałem dla Doktorantów będących we wstępnej fazie prowadzenia badań własnych, ponieważ pracy przyświecało, nieuświadomione i nie wyrażone do tej pory słowami, motto: *złe wiadomości to dobre wiadomości*.

W rozdziale pierwszym przedstawiono krótki rys istniejących metod badania liniowej rozdzielności dwóch zbiorów, nie dyskredytując żadnej z nich. Należy podkreślić, że pomimo iż opisane w tym rozdziale metody, były nieprzydatne z punktu widzenia postawionego sobie celu, to w wielu zastosowaniach są to podejścia, których własności są wystarczające, aby zrealizować z sukcesem różne zadania.

W pracy świadomie zrezygnowano z porównania przedstawionych metod w sposób empiryczny. Znając bowiem ograniczenia każdej z opisanych metod trudno jest znaleźć wspólny „obszar działań", na gruncie którego metody te mogłyby być porównywane. Nie ma wśród tych metod (włącznie z przedstawionym algorytmem LS2S) jednej, najlepszej metody. Ich różnorodność zaś, gwarantuje **możliwość** dokonania przez projektantów najlepszego wyboru w trakcie rozwiązywania jakiegoś szerszego problemu praktycznego, choć nie zwalnia z mądrego przeprowadzenia procesu dokonywania tego wyboru.

Nieskupianie się na porównaniach pozwala dostrzec podobieństwa przedstawionego algorytmu tworzenia hierarchicznego klasyfikatora liniowego do dobrze usystematyzowanych w literaturze drzew decyzyjnych. Oba podejścia łączy wykorzystywana struktura danych, gdzie testy w węzłach są testami prostymi. W przypadku drzew decyzyjnych są to konkretne wartości lub ich przedziały. W przypadku algorytmu LC2S choć nadal mamy do czynienia z testami prostymi, to w rzeczywistości są testy złożone — wartości progowe wyznaczane są jako kombinacje liniowe poszczególnych atrybutów. Natura testów wykorzystywana — dla ustalenia uwagi — w s węzłach drzewa decyzyjnego czy drzewa utworzonego podczas wykonania algorytmu LC2S jest analogiczna do natury atrybutów w zbiorze danych, uzyskanych w czasie obróbki wstępnej danych. Obróbka ta, mająca na celu redukcję liczby cech, może być wykonana dwoma metodami. Pierwszą, gdzie wybrano arbitralnie s atrybutów do opisu każdego z obiektów (pierwotnie, każdy obiekt był opisany przez n cech, $n > s$). Drugą, gdzie do wyboru wspomnianych s cech wykorzystano metodę PCA (*Principal Component Analysis*).

Przeznaczeniem pierwotnym, opisanej w pracy metody tworzenia klasyfikatora hierarchicznego, było wykorzystanie tego klasyfikatora zamiast zbiorów odniesienia w metodzie k najbliższych

sąsiadów. Nie jest to jednak jedyna możliwość zastosowania. Możemy bowiem na tworzony przy wykorzystaniu algorytmu LC2S klasyfikator, spojrzeć nie jak na klasyfikator binarny, lecz trój--decyzyjny: klasa pierwsza–klasa druga–„nie wiem", gdzie decyzja „nie wiem" określona jest przez wyznaczane kolejno, podczas tworzenia klasyfikatora, prześwity ε. Taki klasyfikator może stanowić narzędzie samo w sobie lub być wykorzystanym w konstrukcji narzędzi wspomagających wstępną obróbkę danych.

W rozprawie do wspólnej pracy zostały zaprzęgnięte najpiękniejsze perełki informatyczne, od wspomnianej już pięknej zasady *dziel i zwyciężaj*, po zastosowanie matematyki w celu wykazania poprawności i optymalizacji numerycznej algorytmu.

Liczne fragmenty tej rozprawy zostały opublikowane jako samodzielne artykuły [10, 11, 13, 14], przy czym praca [12] otrzymała pierwszą nagrodę na Międzynarodowych Warsztatach Doktoranckich OWD'05 w kategorii na najlepszy referat[1].

Doświadczenia zebrane do tej pory, związane z testowaniem algorytmu LC2S na rzeczywistych zbiorach danych[2] sugerują istnienie kilku heurystyk zapewniających zoptymalizowanie liczby niezbędnych kroków, wykonywanych w ramach algorytmu LS2S, który jest elementem bazowym algorytmu LC2S, jednak ze względu na aktualnie prowadzone badania w tym zakresie, bedą one źródłem przyszłych publikacji.

[1] *IEE Best Paper Award*.

[2] *Wine Database*, *Car Evaluation Database*, *Iris Plants Database*, *Cancer Database* z zasobów [45].

Dodatek A

Dowód poprawności algorytmu SLS2S

Twierdzenie 1 uzasadnia dołączenie źle sklasyfikowanego punktu do listy rozpinającej. Poprawność algorytmu będzie w pełni udowodniona, gdy wykażemy, że w przypadku zbiorów liniowo rozdzielnych, dodatkowe warunki dotyczące uzyskiwanego rozwiązania w postaci $\max(\varepsilon)$ i $\sum_{i=1}^{n} a_i^2 = 1$ są konieczne i wystarczające, aby rozwiązanie opisywało hiperpłaszczyzny optymalne. Wykażemy także, że takie rozwiązanie jest tylko jedno.

Dowód, ze względów metodycznych, zostanie przeprowadzony dla dwóch przypadków. Podział ten wynika z faktu, że pierwsza część dowodu potwierdza istnienie intuicyjnie poprawnego rozwiązania. Ze względu na podobny charakter obu części dowodu, analiza pierwszego przypadku czyni łatwiejszym w odbiorze drugi przypadek.

W dowodzie przyjęto, że zapis $\sum_{i=p}^{k} x_i$, gdy $p > k$, jest poprawny i wynosi zero. Wielokrotnie wykorzystywana jest także poniższa tożsamość.

Tożsamość 1

$$\left(\sum_{i=1}^{n} x_i\right)^2 = \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_i x_j \tag{A.1}$$

Dowód:

$$\mathbf{L} = \left(\sum_{i=1}^{n} x_i\right)^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j = \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} x_i x_j \overset{*}{=} \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_i x_j =$$

$$= \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_i x_j + 2\sum_{j=n+1}^{n} x_n x_j \overset{**}{=} \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_i x_j = \mathbf{P}$$

$(*)$ ze względu na symetrię wyrażeń $x_i x_j$ i $x_j x_i$.

$(**)$ zgodnie z przyjętą notacją.

1.1. Część pierwsza dowodu — minimalna lista rozpinająca

W tej części dowodu zajmiemy się krokiem $p.4$ dla:

- $p = 0, 1$ oraz
- $p = 2$ przy założeniu, że $\mathrm{card}(K_2^{(1)}) = \mathrm{card}(K_2^{(2)}) = 1$.

Wymienione przypadki oznaczają, że lista rozpinająca hiperpłaszczyzny optymalne jest listą minimalną i po wykonaniu kroków $p.1$, $p.2$ i $p.3$ algorytmu jest spełniony warunek:

$$\mathrm{card}(K \cup K_p^{(1)} \cup K_p^{(2)}) = 2$$

Innymi słowy:

$$K \cup K_p^{(1)} \cup K_p^{(2)} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}\} \text{ gdzie } class(\boldsymbol{y}^{(1)}) \neq class(\boldsymbol{y}^{(2)})$$

Zakładamy również, że i $f^{*-1}(\boldsymbol{y}^{(1)}) \neq f^{*-1}(\boldsymbol{y}^{(2)})$

Bez utraty ogólności możemy założyć, że:

$$class(\boldsymbol{y}^{(1)}) = 1 \;\text{ i }\; class(\boldsymbol{y}^{(2)}) = 2$$

W kroku $p.4$ algorytm SLS2S należy więc wyznaczyć wektor $\boldsymbol{a^*} = [a_1, a_2, \ldots a_n, \varepsilon, a_{n+2}]$, zdefiniowany we wzorach (1.6 lub 3.2), który będzie spełniał następujące równania i warunek: opisane poniżej ograniczenia.

$$\begin{cases} \left\langle \boldsymbol{a}^*, \boldsymbol{y}^{(1)} \right\rangle &= 0 \\ \left\langle \boldsymbol{a}^*, \boldsymbol{y}^{(2)} \right\rangle &= 0 \\ \sum_{i=1}^{n} (a_i^*)^2 &= 1 \\ max(\varepsilon) \end{cases} \tag{A.2}$$

Przy czym, wobec założeń:

$$\begin{array}{rcl} \boldsymbol{a}^* &=& \left[\; a_1^*, \;\ldots,\; a_n^*, \;\; \varepsilon, \; a_{n+2}^*\right] \\ \boldsymbol{y}^{(1)} &=& \left[\; x_1^{(1)}, \;\ldots,\; x_n^{(1)}, \;\; 0, \;\; 1 \;\right] \\ \boldsymbol{y}^{(2)} &=& \left[-x_1^{(2)}, \;\ldots,\; -x_n^{(2)}, \; -1, \; -1 \;\right] \end{array}$$

układ (A.2) możemy zapisać w poniżej przedstawionej postaci:

$$\begin{cases} \sum_{i=1}^{n} a_i^* x_i^{(1)} \qquad\quad + \quad a_{n+2}^* &= 0 \\ -\sum_{i=1}^{n} a_i^* x_i^{(2)} \quad - \quad \varepsilon \quad - \quad a_{n+2}^* &= 0 \\ \sum_{i=1}^{n} (a_i^*)^2 = 1 \\ max(\varepsilon) \end{cases} \tag{A.3}$$

Wyznaczając a_{n+2}^* z pierwszego równania układu (A.3) i podstawiając do drugiego równania, otrzymujemy układ równoważny:

$$\begin{cases} a_{n+2}^* &= -\sum_{i=1}^{n} a_i^* x_i^{(1)} \\ \sum_{i=1}^{n} a_i^* \left(x_i^{(1)} - x_i^{(2)}\right) - \varepsilon &= 0 \\ \sum_{i=1}^{n} (a_i^*)^2 &= 1 \\ max(\varepsilon) \end{cases} \tag{A.4}$$

Dla uproszczenia dalszych obliczeń definiujemy:

$$d_i = x_i^{(1)} - x_i^{(2)} \quad \text{dla} \quad i = 1,\ \ldots,\ n \tag{A.5}$$

i bierzemy $k=1$, jeśli $d_1 \neq 0$ lub takie k, przy którym $d_k \neq 0$, ale $d_i = 0$ dla $0 < i < k$. Wobec przyjętych założeń, takie k istnieje. Wówczas układ (A.4) jest równoważny:

$$\begin{cases} a_{n+2}^* = -\sum_{i=1}^{n} a_i^* x_i^{(1)} \\ a_k^* d_k = -\left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right) \\ \sum_{i=1}^{k-1} (a_i^*)^2 + (a_k^*)^2 + \sum_{i=k+1}^{n} (a_i^*)^2 - 1 \ = \ 0 \\ max(\varepsilon) \end{cases} \tag{A.6}$$

Mnożąc trzecie równanie układu (A.6) przez d_k^2 i następnie podstawiając do niego drugie równanie, otrzymujemy układ:

$$\begin{cases} a_{n+2}^* = -\sum_{i=1}^{n} a_i^* x_i^{(1)} \\ a_k^* d_k = -\left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right) \\ \sum_{i=1}^{k-1} d_k^2 (a_i^*)^2 + \left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right)^2 + \sum_{i=k+1}^{n} d_k^2 a_i^{*2} - d_k^2 \ = \ 0 \\ max(\varepsilon) \end{cases} \tag{A.7}$$

Równanie trzecie układu (A.7) potraktujemy jako równanie kwadratowe ze zmienną a_1^* i parametrami a_2^*, ..., a_n^*, ε. Aby istniało rozwiązanie układu równań (A.7) wyróżnik tego równania musi być nieujemny. Oznacza to, że możemy wyróżnik ten potraktować jako nierówność kwadratową ze zmienną a_i^* i parametrami a_{i+1}^*, ..., a_n^*, ε i obliczyć kolejno dla $i = 2,\ ...,\ k–1$ odpowiednie wyróżniki. Wykorzystując lemat 1 wyznaczamy wszystkie wyróżniki $\Delta_{a_i^*}$ dla $i=1,\ ...,\ k–1$.

Lemma 1 *Traktując równanie postaci:*

$$\sum_{i=1}^{k-1} d_k^2 (a_i^*)^2 + c = 0$$

jako równanie kwadratowe ze zmienną a_i^*, jego kolejno obliczane wyróżniki są równe:

$$\Delta_{a_i^*} = \upsilon_i \left(\sum_{j=i+1}^{k-1} d_k^2 (a_j^*)^2 + c \right)$$

Współczynnik υ_i zdefiniowany jest następująco: $\upsilon_i = -4 d_k^2 \upsilon_{i-1}^2$, gdzie $\upsilon_0 = -1$. Tutaj c jest wyrażeniem nie zawierającym czynnika a_i^* w pierwszej lub drugiej potędze dla $i = 1,\ \ldots,\ k–1$ i może być łatwo wyliczony z trzeciego równania układu (A.7) tzn.:

$$c = \left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right)^2 + \sum_{i=k+1}^{n} d_k^2 a_i^{*2} - d_k^2$$

.

Dowód. Prawdziwość lematu 1 wykażemy przy pomocy indukcji matematycznej. Sprawdźmy poprawność dla i=1. Wyróżnik równania:

$$d_k^2 a_1^{*2} + \sum_{i=2}^{k-1} d_k^2 (a_i^*)^2 + c = 0,$$

w którym jako zmienną potraktujemy a_1^*; równy jest:

$$\Delta = -4d_k^2 \left(\sum_{i=2}^{k-1} d_k^2 (a_i^*)^2 + c \right) = \upsilon_1 \left(\sum_{i=2}^{k-1} d_k^2 (a_i^*)^2 + c \right) = \Delta_{a_1^*}.$$

Zakładamy, że lemat 1 jest prawdziwy dla pewnej liczby s $(1 < s < k-1)$. Oznacza to, że wyróżnik równania ze zmienną a_s^* równy jest:

$$\Delta_{a_s^*} = \upsilon_s \left(\sum_{j=s+1}^{k-1} d_k^2 (a_j^*)^2 + c \right)$$

Traktujemy powyższy wyróżnik jako nierówność ze zmienną a_{s+1}^*:

$$\Delta_{a_s^*} = \upsilon_s \left(d_k^2 (a_{s+1}^*)^2 + \sum_{j=s+2}^{k-1} d_k^2 (a_j^*)^2 + c \right) \geqslant 0$$

Wówczas wyróżnik tej nierówności jest równy:

$$\Delta = -4\upsilon_s^2 d_k^2 \left(\sum_{j=s+2}^{k-1} d_k^2 (a_j^*)^2 + c \right) = \upsilon_{s+1} \left(\sum_{j=s+2}^{k-1} d_k^2 (a_j^*)^2 + c \right) = \Delta_{a_{s+1}^*}.$$

Z prawdziwości lematu dla wartości s wynika prawdziwość dla wartości $s+1$, więc na mocy indukcji matematycznej lemat jest prawdziwy dla $i = 1, \ldots, k-1$. □

Ostatni z wyznaczonych na podstawie lematu 1 wyróżników trzeciego równania układu (A.7) ma postać:

$$\Delta_{a_{k-1}^*} = \upsilon_{k-1} \left(\left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right)^2 + \sum_{i=k+1}^{n} d_k^2 a_i^{*2} - d_k^2 \right) \tag{A.8}$$

Zauważmy, że wszystkie υ_i dla $i = 1, \ldots, k-1$ są niedodatnie. Aby układ (A.7) miał rozwiązanie, wyróżnik (A.8) musi być nieujemny. Możemy więc zapisać ten warunek jako nierówność i potraktować ją jako nierówność kwadratową ze zmienną a_{k+1}^* i parametrami a_{k+2}^*, ..., a_n^*, ε. Analogicznie jak w przypadku składowych a_i^* dla $i = 1, \ldots, k-1$ wyznaczamy kolejne wyróżniki na podstawie lematu 2.

Lemma 2 *Traktując nierówność postaci:*

$$\left(\sum_{i=k+1}^{n} a_i^* d_i - \varepsilon \right)^2 + \sum_{i=k+1}^{n} d_k^2 a_i^{*2} - d_k^2 \leqslant 0$$

jako wyrażenie kwadratowe ze zmienną a_i^* przy $k{+}1 \leqslant i \leqslant n$ (rozwinięcie lewej strony nierówności przedstawiono we wzorze (A.9)), jego kolejno obliczane wyróżniki są równe:

$$\Delta_{a_i^*} = \Upsilon_i \left(\sum_{j=i+1}^{n} \left(\sum_{l=k}^{i} d_l^2 + d_j^2 \right) \left(a_j^*\right)^2 - 2\varepsilon \sum_{j=i+2}^{n} d_j a_j^* + 2 \sum_{j=i+2}^{n-1} \sum_{l=j+1}^{n} d_j d_l a_j^* a_l^* \right) +$$
$$+ \Upsilon_i \left(\varepsilon^2 - \sum_{j=k}^{i} d_j^2 \right)$$

Współczynnik Υ_i zdefiniowany jest następująco: $\Upsilon_i = -4\Upsilon_{i-1}^2 \sum_{j=k}^{i-1} d_j^2$, przy $\Upsilon_k = -1$.

Dowód. Prawdziwość lematu 2 wykażemy przy pomocy indukcji matematycznej. Nierówność z lematu 2 rozwiniemy do postaci:

$$\left(\sum_{i=k+1}^{n} d_i a_i^* - \varepsilon \right)^2 + \sum_{i=k+1}^{n} d_k^2 \left(a_i^*\right)^2 - d_k^2 =$$
$$= \sum_{i=k+1}^{n} d_i^2 \left(a_i^*\right)^2 + \varepsilon^2 + 2 \sum_{i=k+1}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* - 2\varepsilon \sum_{i=k+1}^{n} d_i a_i^* + \sum_{i=k+1}^{n} d_k^2 \left(a_i^*\right)^2 - d_k^2 =$$
$$= \sum_{i=k+1}^{n} \left(d_i^2 + d_k^2 \right) \left(a_i^*\right)^2 + 2 \sum_{i=k+1}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* - 2\varepsilon \sum_{i=k+1}^{n} d_i a_i^* + \varepsilon^2 - d_k^2 \leqslant 0 \qquad \text{(A.9)}$$

Sprawdźmy poprawność dla $i = k{+}1$. Nierówność, w której jako zmienną potraktujemy a_{k+1}^*, ma postać:

$$\left(d_{k+1}^2 + d_k^2 \right) \left(a_{k+1}^*\right)^2 + 2 d_{k+1} \left(\sum_{j=k+2}^{n} d_j a_j^* - \varepsilon \right) a_{k+1}^*$$
$$+ \left(\sum_{i=k+2}^{n} \left(d_i^2 + d_k^2 \right) \left(a_i^*\right)^2 + 2 \sum_{i=k+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* - 2\varepsilon \sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - d_k^2 \right) \leqslant 0$$

Wyróżnik Δ jest równy:

$$\Delta = 4 d_{k+1}^2 \left(\sum_{j=k+2}^{n} d_j a_j^* - \varepsilon \right)^2 - 4 \left(d_{k+1}^2 + d_k^2 \right) \left(\sum_{i=k+2}^{n} \left(d_i^2 + d_k^2 \right) \left(a_i^*\right)^2 \right)$$
$$- 4 \left(d_{k+1}^2 + d_k^2 \right) \left(2 \sum_{i=k+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* - 2\varepsilon \sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - d_k^2 \right)$$

Prześledźmy przekształcenia, które pozwalają uprościć wyrażenie i pokazać prawdziwość lematu 2 dla wartości $k{+}1$:

$$\begin{aligned}
\Delta &= 4d_{k+1}^2\left(\underline{\sum_{i=k+2}^{n} d_i^2\,(a_i^*)^2} \;+\; \underline{\underline{2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*}} \;-\; \underline{\underline{2\varepsilon\sum_{i=k+2}^{n} d_i a_i^*}} \;+\; \underline{\underline{\varepsilon^2}}\right)\\
&\quad -\; 4d_{k+1}^2\left(\underline{\sum_{i=k+2}^{n} d_i^2\,(a_i^*)^2} \;+\; \sum_{i=k+2}^{n} d_k^2\,(a_i^*)^2 + \;\underline{\underline{2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*}}\right)\\
&\quad -\; 4d_{k+1}^2\left(\underline{\underline{-2\varepsilon\sum_{i=k+2}^{n} d_i a_i^*}} \;+\; \underline{\underline{\varepsilon^2}} - d_k^2\right) - 4d_k^2\left(-2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - d_k^2\right)\\
&\quad -\; 4d_k^2\left(\sum_{i=k+2}^{n}\left(d_i^2+d_k^2\right)(a_i^*)^2 + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) =\\
&= -4d_{k+1}^2 d_k^2\left(\sum_{i=k+2}^{n}(a_i^*)^2 - 1\right) - 4d_k^2\left(-2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - d_k^2\right)\\
&\quad -\; 4d_k^2\left(\sum_{i=k+2}^{n}\left(d_i^2+d_k^2\right)(a_i^*)^2 + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) =\\
&= -4d_k^2\left(\sum_{i=k+2}^{n}\underline{d_{k+1}^2\,(a_i^*)^2} - d_{k+1}^2\right) - 4d_k^2\left(-2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - d_k^2\right)\\
&\quad -\; 4d_k^2\left(\sum_{i=k+2}^{n}\underline{\left(d_i^2+d_k^2\right)(a_i^*)^2} + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) =\\
&= -4d_k^2\left(\sum_{i=k+2}^{n}\left(\underbrace{d_k^2+d_{k+1}^2}+d_i^2\right)(a_i^*)^2 + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right)\\
&\quad -\; 4d_k^2\left(-2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + \varepsilon^2 - \underbrace{d_k^2 - d_{k+1}^2}\right) =\\
&= -4(-1)^2\sum_{j=k}^{k} d_j^2\left(\sum_{i=k+2}^{n}\left(\sum_{j=k}^{k+1} d_j^2 + d_i^2\right)(a_i^*)^2 - 2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) \;+\\
&\quad +\; -4(-1)^2\sum_{j=k}^{k} d_j^2\left(\varepsilon^2 - \sum_{j=k}^{k+1} d_j^2\right) = \Delta_{a_{k+1}^*} =\\
&= -4\Upsilon_k^2\sum_{j=k}^{i-1} d_j^2\left(\sum_{i=k+2}^{n}\left(\sum_{j=k}^{k+1} d_j^2 + d_i^2\right)(a_i^*)^2 - 2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) \;+\\
&\quad +\; -4\Upsilon_k^2\sum_{j=k}^{i-1} d_j^2\left(\varepsilon^2 - \sum_{j=k}^{k+1} d_j^2\right) = \Delta_{a_{k+1}^*} =\\
&= \Upsilon_{k+1}\left(\sum_{i=k+2}^{n}\left(\sum_{j=k}^{k+1} d_j^2 + d_i^2\right)(a_i^*)^2 - 2\varepsilon\sum_{i=k+2}^{n} d_i a_i^* + 2\sum_{i=k+2}^{n-1}\sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*\right) \;+\\
&\quad +\; \Upsilon_{k+1}\left(\varepsilon^2 - \sum_{j=k}^{k+1} d_j^2\right) = \Delta_{a_{k+1}^*}.
\end{aligned}$$

Zakładamy, że lemat 2 jest prawdziwy dla pewnej liczby s $(k+1<s<n)$. Oznacza to, że wyróżnik równania ze zmienną a_s^* równy jest:

$$\Delta_{a_s^*} = \Upsilon_s \left(\sum_{i=s+1}^{n} \left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2 - 2\varepsilon \sum_{i=s+1}^{n} d_i a_i^* + 2 \sum_{i=s+1}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) + \\ + \Upsilon_s \left(\varepsilon^2 - \sum_{j=k}^{s} d_j^2 \right)$$

Traktując powyższy wyróżnik jako nierówność ze zmienną a_{s+1}^*:

$$\Delta_{a_s^*} = \Upsilon_s \left(\sum_{j=k}^{s} d_j^2 + d_{s+1}^2 \right) (a_{s+1}^*)^2 + 2\Upsilon_s d_{s+1} \left(\sum_{j=s+2}^{n} d_j a_j^* - \varepsilon \right) a_{s+1}^* + \\ + \Upsilon_s \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2 - 2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* + \varepsilon^2 - \sum_{j=k}^{s} d_j^2 \right)$$

wyróżnik tej nierówności jest równy:

$$\Delta = 4\Upsilon_s^2 d_{s+1}^2 \left(\sum_{i=s+2}^{n} d_i a_i^* - \varepsilon \right)^2 \\ - 4\Upsilon_s^2 \left(\sum_{i=k}^{s} d_i^2 + d_{s+1}^2 \right) \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2 \right) \\ - 4\Upsilon_s^2 \left(\sum_{i=k}^{s} d_i^2 + d_{s+1}^2 \right) \left(2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* - 2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + \varepsilon^2 - \sum_{i=k}^{s} d_i^2 \right)$$

Poniższe przekształcenia pozwalają uprościć wyrażenie i pokazać prawdziwość lematu 2 dla $s+1$:

$$\Delta = 4\Upsilon_s^2 d_{s+1}^2 \left(\underline{\sum_{i=s+2}^{n} d_i^2 (a_i^*)^2} + \underline{\underline{2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*}} - \underline{\underline{2\varepsilon \sum_{i=s+2}^{n} d_i a_i^*}} + \underline{\underline{\underline{\varepsilon^2}}} \right) \\ - 4\Upsilon_s^2 d_{s+1}^2 \left(\sum_{i=s+2}^{n} \sum_{j=k}^{s} d_j^2 (a_i^*)^2 + \underline{\sum_{i=s+2}^{n} d_i^2 (a_i^*)^2} + \underline{\underline{2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^*}} \right) \\ - 4\Upsilon_s^2 d_{s+1}^2 \left(\underline{\underline{-2\varepsilon \sum_{i=s+2}^{n} d_i a_i^*}} + \underline{\underline{\underline{\varepsilon^2}}} - \sum_{i=k}^{s} d_i^2 \right) \\ - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2 + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) \\ - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(-2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + \varepsilon^2 - \sum_{i=k}^{s} d_i^2 \right) =$$

$$
\begin{aligned}
&= -4\Upsilon_s^2 d_{s+1}^2 \sum_{j=k}^{s} d_j^2 \left(\sum_{i=s+2}^{n} (a_i^*)^2 - 1 \right) \\
&\quad - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2 + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) \\
&\quad - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(-2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + \varepsilon^2 - \sum_{i=k}^{s} d_i^2 \right) = \\
&= -4\Upsilon_s^2 \sum_{j=k}^{s} d_j^2 \left(\sum_{i=s+2}^{n} \underline{d_{s+1}^2 (a_i^*)^2} - \underline{\underline{d_{s+1}^2}} \right) \\
&\quad - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(\sum_{i=s+2}^{n} \underline{\left(\sum_{j=k}^{s} d_j^2 + d_i^2 \right) (a_i^*)^2} + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) \\
&\quad - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(-2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + \varepsilon^2 - \underline{\underline{\sum_{i=k}^{s} d_i^2}} \right) = \\
&= -4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s+1} d_j^2 + d_i^2 \right) (a_i^*)^2 + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) \\
&\quad - 4\Upsilon_s^2 \sum_{i=k}^{s} d_i^2 \left(-2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + \varepsilon^2 - \sum_{i=k}^{s+1} d_i^2 \right) = \\
&= -4\Upsilon_{s+1}^2 \left(\sum_{i=s+2}^{n} \left(\sum_{j=k}^{s+1} d_j^2 + d_i^2 \right) (a_i^*)^2 - 2\varepsilon \sum_{i=s+2}^{n} d_i a_i^* + 2 \sum_{i=s+2}^{n-1} \sum_{j=i+1}^{n} d_i d_j a_i^* a_j^* \right) \\
&\quad - 4\Upsilon_{s+1}^2 \left(\varepsilon^2 - \sum_{i=k}^{s+1} d_i^2 \right) = \Delta_{a_{s+1}^*}.
\end{aligned}
$$

Z prawdziwości lematu dla wartości s wynika prawdziwość dla wartości $s+1$, więc na mocy indukcji matematycznej lemat jest prawdziwy dla $i=k+1,\ ...,\ n$. □

Ostatni z możliwych do wyliczenia na podstawie lematu 2 wyróżników trzeciego równania układu (A.7) ma postać:

$$\Delta_{a_n^*} = -4\Upsilon_n^2 \left(\varepsilon^2 - \sum_{i=k}^{n} d_i^2 \right)$$

Aby istniało rozwiązanie układu (A.7), wyróżnik ten musi być nieujemny, co oznacza, że:

$$\varepsilon^2 - \sum_{i=k}^{n} d_i^2 \leqslant 0$$

Spełnienie warunku $\max(\varepsilon)$ z układu równań (A.7) pozwala wyznaczyć wartość ε równą:

$$\varepsilon = \sqrt{\sum_{i=k}^{n} d_i^2}$$

Tym samym $\Delta_{a_n^*}$=0.

Wyjściowy układ równań (A.2) a dokładnie jego równoważna postać (A.7) wobec powyższego faktu, że $\Delta_{a_n^*}$=0, jak i wobec faktu, że wszystkie wyróżniki $\Delta_{a_i^*}$ dla $i=1, \ldots, k$–$1, k$+$1, \ldots, n$ traktowane jako wyrażenia kwadratowe mają ujemne współczynniki przy drugiej potędze niewiadomej a_i^*, jest układem, który ma dokładnie jedno rozwiązanie, co należało wykazać. Kończy to pierwszą część dowodu.

Wartość parametru ε wyznaczona w pierwszej części dowodu jest zgodna z intuicją. Maksymalna odległość między hiperpłaszczyznami rozdzielającymi nie może być większa niż odległość między dwoma punktami, z których każdy leży na odpowiedniej hiperpłaszczyźnie.

1.2. Część druga dowodu dla $p = 2, \ldots, n+1$

W tej części dowodu zajmiemy się krokiem p.4 dla:

- $2<p<n+1$ oraz
- $p=2$ przy założeniu, że albo $\mathrm{card}(K_2^{(1)})=2$, albo $\mathrm{card}(K_2^{(2)})=2$.

Po wykonaniu kroków p.1, p.2 i p.3 algorytmu wymienione przypadki oznaczają spełnienie jednego z dwóch warunków:

$$\mathrm{card}(K \cup K_p^{(1)} \cup K_p^{(2)}) = p, \text{ gdy } \mathrm{card}(K) = \emptyset$$

lub

$$\mathrm{card}(K \cup K_p^{(1)} \cup K_p^{(2)}) = p{+}1, \text{ gdy } \mathrm{card}(K) = 1$$

Bez utraty ogólności możemy przyjąć, że lista utworzona ze zbioru $K \cup K_p^{(1)} \cup K_p^{(2)}$ zawiera:

- p_1 punktów, które oryginalnie reprezentują zbiór X_1, czyli $class(\boldsymbol{y}_i) = 1$ dla $i=1, \ldots, p_1$ oraz
- p_2 punktów, które oryginalnie reprezentują zbiór X_2, innymi słowy oznacza to, że $class(\boldsymbol{y}_i) = 2$ dla $i=p_1{+}1, \ldots, (p_1{+}p_2)$.

Należy więc pokazać, że poniższy układ równań ma dokładnie jedno rozwiązanie:

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_1 \rangle &= 0 \\ & \vdots \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_{p_1} \rangle &= 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_{p_1+1} \rangle &= 0 \\ & \vdots \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_{p_1+p_2} \rangle &= 0 \\ \sum_{i=1}^{n} (a_i^*)^2 &= 1 \\ max(\varepsilon) \end{cases} \tag{A.10}$$

Przy czym, wobec przyjętych założeń układ ten możemy zapisać w poniżej przedstawionej postaci:

$$\left\{\begin{array}{l}
\begin{bmatrix}
y_{1,1} & \cdots & y_{1,p_1+p_2} & \cdots & y_{1,n} & 0 & 1 \\
y_{2,1} & \cdots & y_{2,p_1+p_2} & \cdots & y_{2,n} & 0 & 1 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
y_{p_1,1} & \cdots & y_{p_1,p_1+p_2} & \cdots & y_{p_1,n} & 0 & 1 \\
y_{p_1+1,1} & \cdots & y_{p_1+1,p_1+p_2} & \cdots & y_{p_1+1,n} & -1 & -1 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
y_{p_1+p_2,1} & \cdots & y_{p_1+p_2,p_1+p_2} & \cdots & y_{p_1+p_2,n} & -1 & -1
\end{bmatrix}
\begin{bmatrix} a_1^* \\ a_2^* \\ \vdots \\ a_n^* \\ \varepsilon \\ a_{n+2}^* \end{bmatrix} = 0 \\
\sum_{i=1}^{n} (a_i^*)^2 = 1 \\
\max(\varepsilon)
\end{array}\right. \tag{A.11}$$

Z pierwszego równania układu (A.11) wyznaczamy niewiadomą a_{n+2}^*. Umożliwi to wyrugowanie jej z pozostałych równań, ponieważ nie jest ona uwikłana w równanie stopnia drugiego. Z kolejnych $(p_1+p_2)-1$ równań metodą eliminacji Gaussa wyznaczamy $(p_1+p_2)-1$ współczynników poszukiwanego wektora $\boldsymbol{a}^*$, czyli a_i^*, dla $i = 1, \ldots, (p_1+p_2)-1$. Otrzymujemy wówczas, zapisany symbolicznie, układ równań:

$$\left\{\begin{array}{l}
\begin{bmatrix}
y_{1,1} & y_{1,2} & \cdots & y_{1,p_1+p_2-1} & y_{1,p_1+p_2} & \cdots & y_{1,n} & 0 & 1 \\
1 & y_{2,2}^* & \cdots & y_{2,p_1+p_2-1}^* & y_{2,p_1+p_2}^* & \cdots & y_{2,n}^* & y_{2,n+1}^* & 0 \\
0 & 1 & \cdots & y_{3,p_1+p_2-1}^* & y_{3,p_1+p_2}^* & \cdots & y_{3,n}^* & y_{3,n+1}^* & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & 1 & y_{p_1+p_2,p_1+p_2}^* & \cdots & y_{p_1+p_2,n}^* & y_{p_1+p_2,n+1}^* & 0
\end{bmatrix}
\begin{bmatrix} a_1^* \\ a_2^* \\ \vdots \\ a_n^* \\ \varepsilon \\ a_{n+2}^* \end{bmatrix} = 0 \\
\sum_{i=1}^{n} (a_i^*)^2 = 1 \\
\max(\varepsilon)
\end{array}\right. \tag{A.12}$$

w którym wyrazy nowej macierzy oznaczono gwiazdkami. Dla zachowania struktury trójkątnej macierzy układu, dokonano niezbędnych przestawień wierszy.

Stosując metodę podstawień wyznaczonych składowych a_i^* dla $i = 2, \ldots, p_1+p_2-1$, do odpowiednich równań, otrzymujemy p_1+p_2-1 pierwszych współczynników w zależności od pozostałych współczynników: ε i a_i^*, dla $i = (p_1+p_2), \ldots, n$. Symbolicznie zaznaczono to w układzie (A.13). Równanie stopnia drugiego możemy rozdzielić na współczynniki dotychczas wyznaczone (grupa I) i na te, o których nic jeszcze nie wiemy (grupa II):

$$
\begin{cases}
\begin{bmatrix}
y_{1,1} & y_{1,2} & \dots & y_{1,\,p_1+p_2-1} & y_{1,\,p_1+p_2} & \dots & y_{1,\,n} & 0 & 1 \\
1 & 0 & \dots & 0 & y^{**}_{2,\,p_1+p_2} & \dots & y^{**}_{2,\,n} & y^{**}_{2,\,n+1} & 0 \\
0 & 1 & \dots & 0 & y^{**}_{3,\,p_1+p_2} & \dots & y^{**}_{3,\,n} & y^{**}_{3,\,n+1} & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 1 & y^{**}_{p_1+p_2,\,p_1+p_2} & \dots & y^{**}_{p_1+p_2,\,n} & y^{**}_{p_1+p_2,\,n+1} & 0
\end{bmatrix}
\begin{bmatrix} a^*_1 \\ a^*_2 \\ \vdots \\ a^*_n \\ \varepsilon \\ a^*_{n+2} \end{bmatrix} = 0 \\[2ex]
\underbrace{\sum_{i=1}^{p_1+p_2-1} (a^*_i)^2}_{\text{grupa I}} + \underbrace{\sum_{i=p_1+p_2}^{n} (a^*_i)^2}_{\text{grupa II}} = 1 \\[2ex]
\max(\varepsilon)
\end{cases}
\tag{A.13}
$$

Jeśli punkty $\boldsymbol{y}_i$ dla $i=1, \dots, (p_1+p_2)$ nie rozpinają podprzestrzeni $E^{(p_1+p_2)}$, to wykonane operacje określają wartość czynnika $\varepsilon=0$. Oznacza to, że wektor $\boldsymbol{a}^*$ istnieje i można go wyznaczyć, stosując np. ortogonalizację Grama-Schmidta. W przeciwnym wypadku wyznaczone z równania macierzowego układu (A.13) współczynniki a^*_i zapiszemy, używając następującej notacji macierzowej dla przedstawienia iloczynu skalarnego:

$$
a^*_i = -\begin{bmatrix} a^*_{p_1+p_2} & a^*_{p_1+p_2+1} & \dots & a^*_n & \varepsilon \end{bmatrix} \boldsymbol{v}_i \quad \text{dla } i = 1,\ 2,\ \dots,\ p_1+p_2-1 \tag{A.14}
$$

gdzie $\boldsymbol{v}_i^{\mathrm{T}} = [\ y^{**}_{i+1,\,p_1+p_2} \ \ \dots \ \ y^{**}_{i+1,\,n} \ \ y^{**}_{i+1,\,n+1}\ ]$.

Wykorzystując notację (A.14) zapisujemy równanie stopnia drugiego, z rozwiązywanego układu (A.13), w postaci formy kwadratowej następującej postaci:

$$
\begin{bmatrix} a^*_{p_1+p_2} & a^*_{p_1+p_2+1} & \cdots & a^*_n & \varepsilon & 1 \end{bmatrix}
\begin{bmatrix} \\ & \boldsymbol{S}^{(0)} & \\ \\ \end{bmatrix}
\begin{bmatrix} a^*_{p_1+p_2} \\ a^*_{p_1+p_2+1} \\ \cdots \\ a^*_n \\ \varepsilon \\ 1 \end{bmatrix} = 0 \tag{A.15}
$$

gdzie macierz $\boldsymbol{S}^{(0)}$ powstała z podstawienia do równania (A.13) wyznaczonych dotychczas (p_1+p_2-1) pierwszych współczynników a^*_i:

$$
\boldsymbol{S}^{(0)} = \underbrace{\begin{bmatrix} \boxed{\ \boldsymbol{V}\ } & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \begin{matrix} 0 & \cdots & 0 \end{matrix} & 0 \end{bmatrix}}_{\text{grupa I}}
+ \underbrace{\begin{bmatrix} \boxed{\ \boldsymbol{I}\ } & \begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{matrix} & \begin{matrix} 0 & 0 \\ 0 & -1 \end{matrix} \end{bmatrix}}_{\text{grupa II}} \tag{A.16}
$$

gdzie macierz $\boldsymbol{V} = \sum_{i=1}^{p_1+p_2-1} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}}$.

Macierz $\boldsymbol{I}$ odpowiada za drugi składnik sumy równania kwadratowego układu (A.13). Jest to macierz jednostkowa o wymiarach: $(n-(p_1+p_2)+1) \times (n-(p_1+p_2)+1)$. Wartość -1 umieszczona w skrajnym, prawym, dolnym elemencie macierzy będącej drugim czynnikiem sumy, odpowiada przeniesionej 1 z prawej na lewą stronę równania stopnia drugiego układu (A.13).

Przedstawiona równoważna postać (A.15) równania stopnia drugiego z układu (A.13) umożliwia sprawne potraktowanie tego równania jako równania kwadratowego kolejno z niewiadomą $a^*_{p_1+p_2}$, $a^*_{p_1+p_2+1}$, $\ldots$, aż do a^*_n, podobnie jak w pierwszej części dowodu.

W kontekście rozwiązywanego równania kwadratowego z niewiadomą $a^*_{p_1+p_2}$ wyodrębnimy w macierzy $\boldsymbol{S}^{(0)}$ składniki odpowiadające (szkolnej) postaci równania kwadratowego $ax^2+bx+c=0$, którą zapiszemy w postaci formy kwadratowej:

$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = 0 \tag{A.17}$$

W przypadku, gdy równanie kwadratowe zawiera dodatkowe czynniki z m parametrami M_i dla $(i=1, 2, \ldots, m)$, w co najwyżej drugiej potędze, formę kwadratową (A.17) będziemy mogli zapisać w postaci:

$$\begin{bmatrix} x & M_1 & \ldots & M_m & 1 \end{bmatrix} \begin{bmatrix} a & \frac{1}{2}\boldsymbol{b}^{\mathrm{T}} \\ \frac{1}{2}\boldsymbol{b} & \boldsymbol{c} \end{bmatrix} \begin{bmatrix} x \\ M_1 \\ \ldots \\ M_m \\ 1 \end{bmatrix} = 0 \tag{A.18}$$

gdzie tylko a pozostaje skalarem, zaś $\frac{1}{2}\boldsymbol{b}$ jest wektorem o $(m+1)$ składowych, zaś $\boldsymbol{c}$ macierzą o wymiarach $(m+1)\times(m+1)$.

Porównanie struktury równań (A.15) i (A.18) pozwala na wyodrębnienie odpowiednich składników w macierzy $\boldsymbol{S}^{(0)}$:

$$\boldsymbol{S}^{(0)} = \begin{bmatrix} a^{(0)} & \frac{1}{2}\boldsymbol{b}^{(0)\,\mathrm{T}} \\ \frac{1}{2}\boldsymbol{b}^{(0)} & \boldsymbol{c}^{(0)} \end{bmatrix} \tag{A.19}$$

Wyróżnik równania kwadratowego (A.15) z pierwszą niewiadomą $a^*_{p_1+p_2}$ jest równy wyrażeniu, które zapisane jako forma kwadratowa ma następującą postać:

$$\Delta_{a^*_{p_1+p_2}} = \begin{bmatrix} a^*_{p_1+p_2+1} & a^*_{p_1+p_2+2} & \cdots & a^*_n & \varepsilon & 1 \end{bmatrix} \begin{bmatrix} & & & \\ & \boldsymbol{S}^{(1)} & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} a^*_{p_1+p_2+1} \\ a^*_{p_1+p_2+2} \\ \vdots \\ a^*_n \\ \varepsilon \\ 1 \end{bmatrix}$$

Macierz $\boldsymbol{S}^{(1)}$ wyliczono z następującego wzoru dla $i=1$:

$$\boldsymbol{S}^{(i)} = \boldsymbol{b}^{(i-1)}\boldsymbol{b}^{(i-1)^{\mathrm{T}}} - 4a^{(i-1)}\boldsymbol{c}^{(i-1)} = 4\left(\boldsymbol{b}^{(i-1)^*}\boldsymbol{b}^{(i-1)^{*^{\mathrm{T}}}} - a^{(i-1)}\boldsymbol{c}^{(i-1)}\right) \tag{A.20}$$

Przyjmując, że każda z macierzy $\boldsymbol{S}^{(i)}$ ma strukturę macierzy $\boldsymbol{S}^{(0)}$ z równania (A.19), wektor $\boldsymbol{b}^{(i)^*}$ jest określony następująco:

$$\boldsymbol{b}^{(i)^*} = \tfrac{1}{2}\boldsymbol{b}^{(i)}.$$

Macierze $\boldsymbol{S}^{(i)}$ dla $i=1, 2, \ldots, n-(p_1+p_2)+1$ są macierzami o wymiarach $(n-(p_1+p_2)+3-i)\times(n-(p_1+p_2)+3-i)$.

Wyróżnik $\Delta_{a^*_{p_1+p_2}}$ ma być nieujemny, jeśli układ wyjściowy (A.10) ma mieć tylko jedno rozwiązanie. Nierówność $\Delta_{a^*_{p_1+p_2}} \geqslant 0$ potraktujemy, tak jak w pierwszej części dowodu, jako nierówności kwadratowe kolejno z niewiadomą $a^*_{p_1+p_2+i}$ dla $i=1, 2, \ldots, n-(p_1+p_2)$. Wyróżniki tych nierówności uzyskamy stosując następujący wzór:

$$\Delta_{a^*_{p_1+p_2+i}} = \begin{bmatrix} a^*_{p_1+p_2+i+1} & \cdots & a^*_n & \varepsilon & 1 \end{bmatrix} \begin{bmatrix} & & & \\ & \boldsymbol{S}^{(i+1)} & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} a^*_{p_1+p_2+i+1} \\ \cdots \\ a^*_n \\ \varepsilon \\ 1 \end{bmatrix} \tag{A.21}$$

Układ wyjściowy(A.10) ma dokładnie jedno rozwiązanie, jeśli wykażemy prawdziwość przedstawionych poniżej dwóch tez, które dotyczą wartości niektórych elementów macierzy $\boldsymbol{S}^{(i)}$ dla $i=1, \ldots, n-(p_1+p_2)+1$. Prawdziwość tych tez oznaczać będzie, że ostatnia do rozwiązania nierówność mająca postać:

$$\Delta_{a^*_n} = \begin{bmatrix} \varepsilon & 1 \end{bmatrix} \left[\boldsymbol{S}^{(n-(p_1+p_2)+1)}\right] \begin{bmatrix} \varepsilon \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon & 1 \end{bmatrix} \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} \begin{bmatrix} \varepsilon \\ 1 \end{bmatrix} = \delta_1\varepsilon^2 + \delta_2 \geqslant 0 \tag{A.22}$$

pozwala w sposób jednoznaczny wyznaczyć maksymalną wartość ε ponieważ δ_1 jest wartością ujemną, natomiast δ_2 jest wartością nieujemną.

Teza 1 *Współczynniki przy drugiej potędze niewiadomej $a^*_{p_1+p_2+i}$ są ujemne, w kolejno rozwiązywanych nierówności kwadratowych: $\Delta_{a^*_{p_1+p_2+i}} \geqslant 0$ dla $i=0, 1, \ldots, n-(p_1+p_2)$ są ujemne. Innymi słowy, każdy skrajny lewy, górny element $s^{(i+1)}_{1,1}$ macierzy $\boldsymbol{S}^{(i+1)}$ jest ujemny.*

Teza 2 *Skrajny, dolny, prawy element* $s_{z,z}^{(i+1)}$ *każdej z macierzy* $\boldsymbol{S}^{(i+1)}$ *dla* $i=0,\ 1,\ \ldots,\ n-(p_1+p_2)$ *jest nieujemny. Wartość* $z=n-(p_1+p_2)+3-(i+1)$.

W dowodach poprawności tez (1) i (2) wykorzystamy następujące własności składowych macierzy $\boldsymbol{S}^{(0)}$ z równania (A.16).

Własność 1 *Z faktu, że macierz* $\boldsymbol{V}$ *jest macierzą dodatnio określoną wynika, że wszystkie minory główne tej macierzy są dodatnie.*

Definicja 7 *Niech macierze* $\boldsymbol{C}_i$ *będą podmacierzami macierzy* $\boldsymbol{S}^{(0)}$ *uzyskanymi przez skreślenie z niej wszystkich wierszy i kolumn, których indeksy są większe od wartości* i *dla* $i=1,\ \ldots,\ ((n-1)-(p_1+p_2)+2)$.

Własność 2 *Dzięki własności 1 wszystkie elementy znajdujące na przekątnych macierzy* $\boldsymbol{C}_i$ *dla* $i=1,\ \ldots,\ ((n-1)-(p_1+p_2)+2)$ *są dodatnie.*

Własność 3 *Dzięki własności 1 i twierdzeniu o wielomianie charakterystycznym, wyznaczniki wszystkich macierzy* $\boldsymbol{C}_i$ *dla* $i=1,\ 2,\ \ldots,\ n-(p_1+p_2)+2$ *są dodatnie.*

Lemma 3 *Elementy* $s_{ij}^{(k)}$ *macierzy* $\boldsymbol{S}^{(k)}$ *dla* $k=1,\ 2,\ \ldots\ n-(p_1+p_2)+1$ *mają wartości określone przez poniższy wzór:*

$$s_{ij}^{(k)}=\Psi_k \det\begin{bmatrix} & & & s_{1,j+k}^{(0)} \\ & \boldsymbol{C}_k & & s_{2,j+k}^{(0)} \\ & & & \vdots \\ & & & s_{k,j+k}^{(0)} \\ s_{i+k,1}^{(0)} & s_{i+k,2}^{(0)} & \cdots & s_{i+k,j+k}^{(0)} \end{bmatrix} \tag{A.23}$$

Współczynnik Ψ_k *zdefiniowany jest następująco:* $\Psi_k=-4\Psi_{k-1}^2 \det[\boldsymbol{C}_{k-1}]$, *przy* $\Psi_1=-4$.

Dowód: Prawdziwość lematu 3 wykażemy przy pomocy indukcji matematycznej. Na podstawie wzoru (A.20) możemy wyznaczyć wartość każdego elementu $s_{ij}^{(k)}$ macierzy $\boldsymbol{S}^{(k)}$ dla $k=1,\ 2,\ \ldots\ n-(p_1+p_2)+1$. Uwzględniając, na podstawie struktury macierzy $S^{(k-1)}$, położenie w niej skalara $a^{(k-1)}$, wektora $\frac{1}{2}\boldsymbol{b}^{(k-1)}$ i macierzy $\boldsymbol{c}^{(k-1)}$, otrzymujemy następujący wzór:

$$s_{ij}^{(k)}=4\left(\tfrac{1}{2}\boldsymbol{b}_i^{(k-1)}\tfrac{1}{2}\boldsymbol{b}_j^{(k-1)}-a^{(k-1)}\boldsymbol{c}_{i,j}^{(k-1)}\right)=4\left(s_{i+1,1}^{(k-1)}s_{1,j+1}^{(k-1)}-s_{1,1}^{(k-1)}s_{i+1,j+1}^{(k-1)}\right) \tag{A.24}$$

Korzystając z wyprowadzonego wzoru (A.24) sprawdźmy poprawność lematu 3 dla k=1:

$$\mathbf{L}=s_{ij}^{(1)}=4\left(s_{i+1,1}^{(0)}s_{1,j+1}^{(0)}-s_{1,1}^{0)}s_{i+1,j+1}^{(0)}\right)=-4\det\begin{bmatrix} s_{1,1}^{(0)} & s_{1,j+1}^{(0)} \\ s_{i+1,1}^{(0)} & s_{i+1,j+1}^{(0)} \end{bmatrix}\overset{(*)}{=}\Psi_1\det\begin{bmatrix} \boldsymbol{C}_1 & s_{1,j+1}^{(0)} \\ s_{i+1,1}^{(0)} & s_{i+1,j+1}^{(0)} \end{bmatrix}=\mathbf{P}$$

Równość $\overset{(*)}{=}$ zachodzi na podstawie definicji macierzy $\boldsymbol{C}_i$ i współczynnika Ψ_i dla $i=1$.

Zakładamy, że lemat 3 jest prawdziwy dla pewnego m, gdzie $1<m<n-(p_1+p_2)+1$.

Wykażemy, że z prawdziwości lematu 3 dla wartości m wynika prawdziwość dla $m{+}1$. Obliczając różnicę $\Delta_s = s_{i,j}^{(m+1)} - s_{i,j}^{(m+1)}$, gdzie pierwszy składnik liczony jest ze wzoru definiowanego przez lemat 3, zaś drugi z wyprowadzonego wzoru (A.24), wykażemy, że jest ona równa zero.

$$\Delta_s = s_{i,j}^{(m+1)} - s_{i,j}^{(m+1)}$$

$$= \Psi_{(m+1)} \det \begin{bmatrix} \boldsymbol{C}_{m+1} & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m+1,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{i+m+1,m+1} \end{matrix} & s^{(0)}_{i+m+1,j+m+1} \end{bmatrix} - 4\left(s^{(m)}_{i+1,1}s^{(m)}_{1,j+1} - s^{(m)}_{1,1}s^{(m)}_{i+1,j+1}\right) =$$

$$= -4\Psi_m^2 \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_{m+1} & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m+1,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m+1} \end{matrix} & s^{(0)}_{i+m+1,j+m+1} \end{bmatrix}$$

$$-4\Psi_m^2 \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & s^{(0)}_{i+m+1,m+1} \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & s^{(0)}_{m+1,j+m+1} \end{bmatrix}$$

$$+4\Psi_m^2 \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & s^{(0)}_{m+1,m+1} \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & s^{(0)}_{i+m+1,j+m+1} \end{bmatrix}$$

Ponieważ Ψ_m jest wartością z definicji różną od zera, dzielimy otrzymane wyrażenie przez $4\Psi_m^2$. Macierz $\boldsymbol{C}_{m+1}$ przedstawiamy jako macierz zawierającą macierz $\boldsymbol{C}_m$, a z pozostałych wyznacz-

ników przy pomocy rozwinięcia Laplace'a usuwamy skrajny, dolny element:

$$\Delta_s = -\det[\boldsymbol{C}_m]\det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \\ s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} s^{(0)}_{m+1,m+1} & s^{(0)}_{m+1,j+m+1} \\ s^{(0)}_{i+m+1,m+1} & s^{(0)}_{i+m+1,j+m+1} \end{matrix} \end{bmatrix}$$

$$-\left(s^{(0)}_{i+m+1,m+1}\det[\boldsymbol{C}_m] + \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}\right)\cdot$$

$$\cdot\left(s^{(0)}_{m+1,j+m+1}\det[\boldsymbol{C}_m] + \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix}\right)$$

$$+\left(s^{(0)}_{m+1,m+1}\det[\boldsymbol{C}_m] + \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix}\right)\cdot$$

$$\cdot\left(s^{(0)}_{i+m+1,j+m+1}\det[\boldsymbol{C}_m] + \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}\right)$$

Następnie stosujemy rozwinięcie Laplace'a usuwając skrajny, dolny element macierzy w pierwszym składniku otrzymanego wyrażenia, oraz wymnażamy pozostałe składniki.

$$
\begin{aligned}
\Delta_s = & -s^{(0)}_{i+m+1,j+m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & s^{(0)}_{m+1,m+1} \end{bmatrix} \\
& - \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \\ s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} s^{(0)}_{m+1,m+1} & s^{(0)}_{m+1,j+m+1} \\ s^{(0)}_{i+m+1,m+1} & 0 \end{matrix} \end{bmatrix} \\
& - s^{(0)}_{i+m+1,m+1} s^{(0)}_{m+1,j+m+1} \left(\det[\boldsymbol{C}_m]\right)^2 + s^{(0)}_{m+1,m+1} s^{(0)}_{i+m+1,j+m+1} \left(\det[\boldsymbol{C}_m]\right)^2 \\
& - s^{(0)}_{i+m+1,m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& - s^{(0)}_{m+1,j+m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& - \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& + s^{(0)}_{i+m+1,j+m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix}
\end{aligned}
$$

$$+ s^{(0)}_{m+1,m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}$$

$$+ \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}$$

Jeszcze raz stosujemy rozwinięcie Laplace'a usuwając skrajny, dolny element macierzy w pierwszym składniku otrzymanego wyrażenia. W wyniku tej operacji zaznaczone odpowiednio składniki zredukują się.

$$
\begin{aligned}
\Delta_s = & \underline{-s^{(0)}_{i+m+1,j+m+1} s^{(0)}_{m+1,m+1} \left(\det\left[\boldsymbol{C}_m\right]\right)^2} \\
& - s^{(0)}_{i+m+1,j+m+1} \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& - \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \\ s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} s^{(0)}_{m+1,m+1} & s^{(0)}_{m+1,j+m+1} \\ s^{(0)}_{i+m+1,m+1} & 0 \end{matrix} \end{bmatrix} \\
& - s^{(0)}_{i+m+1,m+1} s^{(0)}_{m+1,j+m+1} \left(\det\left[\boldsymbol{C}_m\right]\right)^2 + \underline{s^{(0)}_{m+1,m+1} s^{(0)}_{i+m+1,j+m+1} \left(\det\left[\boldsymbol{C}_m\right]\right)^2} \\
& - s^{(0)}_{i+m+1,m+1} \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& - s^{(0)}_{m+1,j+m+1} \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& - \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
& + s^{(0)}_{i+m+1,j+m+1} \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix}
\end{aligned}
$$

$$+ s^{(0)}_{m+1,m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}$$

$$+ \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}$$

Ponownie stosujemy rozwinięcie Laplace'a usuwając element znajdujący się w kolumnie $m{+}1$ i wierszu $m{+}1$ z pierwszego nieredukującego się składnika. W wyniku tej operacji zredukują się kolejne składniki.

$$\Delta_s = \underbrace{- s^{(0)}_{m+1,m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}}$$

$$- \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \\ s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} 0 & s^{(0)}_{m+1,j+m+1} \\ s^{(0)}_{i+m+1,m+1} & 0 \end{matrix} \end{bmatrix}$$

$$- s^{(0)}_{i+m+1,m+1} s^{(0)}_{m+1,j+m+1} \left(\det[\boldsymbol{C}_m]\right)^2$$

$$- s^{(0)}_{i+m+1,m+1} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix}$$

$$
\begin{aligned}
&- s^{(0)}_{\cancel{m+1},\cancel{j+m+1}} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{i+m+1},1} & \cdots & s^{(0)}_{\cancel{i+m+1},m} \end{matrix} & 0 \end{bmatrix} \\
&- \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{i+m+1},1} & \cdots & s^{(0)}_{\cancel{i+m+1},m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{j+m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{j+m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{m+1},1} & \cdots & s^{(0)}_{\cancel{m+1},m} \end{matrix} & 0 \end{bmatrix} \\
&\underbrace{+ s^{(0)}_{\cancel{m+1},\cancel{m+1}} \det[\boldsymbol{C}_m] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{j+m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{j+m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{i+m+1},1} & \cdots & s^{(0)}_{\cancel{i+m+1},m} \end{matrix} & 0 \end{bmatrix}} \\
&+ \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{m+1},1} & \cdots & s^{(0)}_{\cancel{m+1},m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{j+m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{j+m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{i+m+1},1} & \cdots & s^{(0)}_{\cancel{i+m+1},m} \end{matrix} & 0 \end{bmatrix}
\end{aligned}
$$

Analogicznie, po raz ostatni stosując rozwinięcie Laplace'a, usuwamy z pierwszego nieznikającego składnika otrzymanego wyrażenia elementy znajdujące się w kolumnie $m+1$ i wierszu $m+2$ oraz w kolumnie $m+2$ i wierszu $m+1$.

$$
\Delta_s = s^{(0)}_{\cancel{i+m+1},\cancel{m+1}} \det[\boldsymbol{C}_m] \left(\underline{s^{(0)}_{\cancel{m+1},\cancel{j+m+1}} \det[\boldsymbol{C}_m]} + \underline{\underline{\det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,\cancel{j+m+1}} \\ \vdots \\ s^{(0)}_{m,\cancel{j+m+1}} \end{matrix} \\ \begin{matrix} s^{(0)}_{\cancel{m+1},1} & \cdots & s^{(0)}_{\cancel{m+1},m} \end{matrix} & 0 \end{bmatrix}}} \right)
$$

$$
\begin{aligned}
&+ s^{(0)}_{m+1,j+m+1} \det\left[\boldsymbol{C}_m\right] \det \underbrace{\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}}_{} \\
&- \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \\ s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} \\
&\underline{- s^{(0)}_{i+m+1,m+1} s^{(0)}_{m+1,j+m+1} \left(\det\left[\boldsymbol{C}_m\right]\right)^2} \\
&- s^{(0)}_{i+m+1,m+1} \det\left[\boldsymbol{C}_m\right] \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
&\underline{\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
&- s^{(0)}_{m+1,j+m+1} \det\left[\boldsymbol{C}_m\right] \det \underbrace{\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}}_{} \\
&- \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \\
&+ \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & 0 \end{bmatrix} \det \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & 0 \end{bmatrix}
\end{aligned}
$$

Ostatecznie w wyniku przeprowadzenia powyższych operacji otrzymamy wyrażenie:

$$\begin{aligned}
\Delta_s = &- \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & \begin{matrix} 0 & 0 \end{matrix} \\ \boxed{\begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix}} & \boxed{\begin{matrix} 0 & 0 \end{matrix}} \end{bmatrix} \det[\boldsymbol{C}_m] \\
&- \det\begin{bmatrix} \boxed{\boldsymbol{C}_m} & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & \begin{matrix} 0 \end{matrix} \end{bmatrix} \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \boxed{\begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix}} & \boxed{\begin{matrix} 0 \end{matrix}} \end{bmatrix} \\
&+ \det\begin{bmatrix} \boxed{\boldsymbol{C}_m} & \begin{matrix} s^{(0)}_{1,m+1} \\ \vdots \\ s^{(0)}_{m,m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & \begin{matrix} 0 \end{matrix} \end{bmatrix} \det\begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,j+m+1} \\ \vdots \\ s^{(0)}_{m,j+m+1} \end{matrix} \\ \boxed{\begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix}} & \boxed{\begin{matrix} 0 \end{matrix}} \end{bmatrix}
\end{aligned} \tag{A.25}$$

Aby każdy ze składników otrzymanego wyrażenia był iloczynem wyznaczników macierzy o wymiarach $(m+1)\times(m+1)$ i $m\times m$ zastosujemy rozwinięcie Laplace'a względem zaznaczonych wierszy.

Najpierw jednak zapiszemy wyrażenie (A.25), stosując następującą notację: dla każdej macierzy $\boldsymbol{A}$ zapis $\boldsymbol{A}^{c_1,\ldots,c_n}_{r_1,\ldots,r_n}$ oznaczać będzie macierz powstałą z macierzy $\boldsymbol{A}$ przez usunięcie z niej wierszy $r_1, \ldots, r_n$ i kolumn $c_1, \ldots, c_n$.

Zdefiniujmy macierz $\boldsymbol{W}$, która pozwoli zapisać wyrażenie (A.25) przy pomocy wspomnianej notacji:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{C}_m & \begin{matrix} s^{(0)}_{1,m+1} & s^{(0)}_{1,j+m+1} \\ \vdots & \vdots \\ s^{(0)}_{m,m+1} & s^{(0)}_{m,j+m+1} \end{matrix} \\ \begin{matrix} s^{(0)}_{m+1,1} & \cdots & s^{(0)}_{m+1,m} \end{matrix} & \begin{matrix} 0 & 0 \end{matrix} \\ \begin{matrix} s^{(0)}_{i+m+1,1} & \cdots & s^{(0)}_{i+m+1,m} \end{matrix} & \begin{matrix} 0 & 0 \end{matrix} \end{bmatrix} \tag{A.26}$$

Wówczas:

$$\Delta_s = -\det \boldsymbol{W} \det \boldsymbol{W}^{m+1,\,m+2}_{m+1,\,m+2} - \det \boldsymbol{W}^{m+1}_{m+2} \det \boldsymbol{W}^{m+2}_{m+1} + \det \boldsymbol{W}^{m+2}_{m+2} \det \boldsymbol{W}^{m+1}_{m+1} \tag{A.27}$$

Po rozwinięciu względem zaznaczonych wierszy i po podzieleniu przez $(-1)^{(m+2+l+1)}$ otrzymamy:

$$\Delta_s = \sum_{l=1}^{m} s^{(0)}_{i+m+1,l} \left(\det \boldsymbol{W}^{l}_{m+2} \det \boldsymbol{W}^{m+1,\,m+2}_{m+1,\,m+2} - \det \boldsymbol{W}^{m+1}_{m+2} \det \boldsymbol{W}^{l,\,m+2}_{m+1,\,m+2} + \det \boldsymbol{W}^{m+2}_{m+2} \det \boldsymbol{W}^{l,\,m+1}_{m+1,\,m+2} \right) \tag{A.28}$$

Pokażemy, że każdy składnik uzyskanej sumy jest równy zero na mocy tożsamości Schweinsa, na której przydatność w dowodzie zwrócił mi uwagę prof. Stefan Paszkowski[1], podając niezbędne odwołania do literatury [3, 37]:

Tożsamość 2 (Schweinsa)

$$\begin{vmatrix} c_1 & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_n & a_{n,1} & \cdots & a_{n,n-1} \end{vmatrix} \begin{vmatrix} b_1 & a_{1,1} & \cdots & a_{1,n-2} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n-1} & a_{n-1,1} & \cdots & a_{n-1,n-2} \end{vmatrix} - \begin{vmatrix} b_1 & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ b_n & a_{n,1} & \cdots & a_{n,n-1} \end{vmatrix} \begin{vmatrix} c_1 & a_{1,1} & \cdots & a_{1,n-2} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n-1} & a_{n-1,1} & \cdots & a_{n-1,n-2} \end{vmatrix}$$
$$+ \begin{vmatrix} a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} \end{vmatrix} \begin{vmatrix} b_1 & c_1 & a_{1,1} & \cdots & a_{1,n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_n & c_n & a_{n,1} & \cdots & a_{n,n-2} \end{vmatrix} = 0$$

Zdefiniujmy macierz $\boldsymbol{W}^*$, która pozwoli zapisać tożsamość Schweinsa przy pomocy używanej notacji:

$$\boldsymbol{W}^* = \begin{bmatrix} b_1 & c_1 & a_{1,1} & \cdots & a_{1,n-2} & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_n & c_n & a_{n,1} & \cdots & a_{n,n-2} & a_{n,n-1} \\ b_{n+1} & c_{n+1} & a_{n+1,1} & \cdots & a_{n+1,n-2} & a_{n+1,n-1} \end{bmatrix} \tag{A.29}$$

Tożsamość Schweinsa przyjmuje wówczas postać (macierz $\boldsymbol{W}^*$ jest macierzą $(n+1)\times(n+1)$):

$$\det \boldsymbol{W}^{*\,1}_{n+1} \det \boldsymbol{W}^{2,\,n+1}_{n,\,n+1} - \det \boldsymbol{W}^{*\,2}_{n+1} \det \boldsymbol{W}^{*\,1,\,n+1}_{n,\,n+1} + \det \boldsymbol{W}^{*\,n+1}_{n+1} \det \boldsymbol{W}^{*\,1,\,2}_{n,\,n+1} = 0 \tag{A.30}$$

Weźmy l-ty składnik sumy określającej Δ_s (A.28):

$$s^{(0)}_{i+m+1,l} \left(\det \boldsymbol{W}^{l}_{m+2} \det \boldsymbol{W}^{m+1,\,m+2}_{m+1,\,m+2} - \det \boldsymbol{W}^{m+1}_{m+2} \det \boldsymbol{W}^{l,\,m+2}_{m+1,\,m+2} + \det \boldsymbol{W}^{m+2}_{m+2} \det \boldsymbol{W}^{l,\,m+1}_{m+1,\,m+2} \right) \tag{A.31}$$

Zdefiniujmy macierz $\boldsymbol{W}^{(l)}$, która powstała z macierzy $\boldsymbol{W}$, przez przestawienie l-tej kolumny na pozycję pierwszej kolumny, zaś $(m+1)$ kolumny na pozycję drugiej kolumny. Wówczas możemy

[1] Instytut Niskich Temperatur i Badań Strukturalnych PAN, Wrocław

zapisać następującą równość:

$$
\begin{aligned}
& s^{(0)}_{i+m+1,l}\left(\det \boldsymbol{W}^{l}_{m+2} \det \boldsymbol{W}^{m+1,\,m+2}_{m+1,\,m+2} - \det \boldsymbol{W}^{m+1}_{m+2} \det \boldsymbol{W}^{l,\,m+2}_{m+1,\,m+2} + \det \boldsymbol{W}^{m+2}_{m+2} \det \boldsymbol{W}^{l,\,m+1}_{m+1,\,m+2}\right) \\
& = s^{(0)}_{i+m+1,l}\left(\det {\boldsymbol{W}^{(l)}}^{1}_{m+2} \det {\boldsymbol{W}^{(l)}}^{2,\,m+2}_{m+1,\,m+2} - \det {\boldsymbol{W}^{(l)}}^{2}_{m+2} \det {\boldsymbol{W}^{(l)}}^{1,\,m+2}_{m+1,\,m+2} + \det {\boldsymbol{W}^{(l)}}^{m+2}_{m+2} \det {\boldsymbol{W}^{(l)}}^{1,\,2}_{m+1,\,m+2}\right) \\
& \overset{(*)}{=} 0
\end{aligned}
\tag{A.32}
$$

Równość $\overset{(*)}{=}$ zachodzi na mocy tożsamości Schweinsa w postaci A.30, gdzie macierz $\boldsymbol{W}^{*} = \boldsymbol{W}^{(l)}$ zaś $n = m+1$. Oznacza to, że z prawdziwości lematu 3 dla wartości m wynika prawdziwość dla wartości $m+1$, więc na mocy indukcji matematycznej lemat jest prawdziwy dla $k = 1, 2, \ldots, n-(p_1+p_2)+1$. □

Lemat 3 jest wystarczający, by stwierdzić prawdziwość tez 1 i 2.

Wartość każdego skrajnego, lewego, górnego elementu $s^{(i+1)}_{1,1}$ macierzy $\boldsymbol{S}^{(i+1)}$ dla $i = 0, 1, \ldots, n-(p_1+p_2)$ jest, zgodnie z lematem 3, równa $s^{(i+1)}_{1,1} = \Psi_{i+1} \det \boldsymbol{C}_{i+2}$. Jest więc ujemna na podstawie definicji w tym lemacie współczynnika Ψ_{i+1} i własności 3.

Wartość każdego elementu skrajnego, prawego, dolnego $s^{(i+1)}_{z,z}$ macierzy $\boldsymbol{S}^{(i+1)}$ dla $i = 0, 1, \ldots, n-(p_1+p_2)$, gdzie $z = n-(p_1+p_2)+3-(i+1)$ jest zgodnie z lematem 3 i wartościami elementów macierzy $\boldsymbol{S}^{(0)}$ ze związku (A.16) równa:

$$
s^{(i+1)}_{z,z} = \Psi_{i+1} \det \begin{bmatrix} \boxed{\boldsymbol{C}_{i+1}} & \begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & \ldots & 0 \end{matrix} & -1 \end{bmatrix} = -\Psi_{i+1} \det \boldsymbol{C}_{i+1}
$$

Oznacza to, że $s^{(i+1)}_{z,z}$ jest dodatnie na podstawie definicji w tym lemacie współczynnika $\Psi_{(i+1)}$ i własności 3.

Prawdziwość tez 1 i 2 oznacza, że ostatnia do rozwiązania nierówność (A.22) staje się równością, tj. $\Delta_{a^{*}_{n}} = 0$, gdy przyjmiemy maksymalną wartość niewiadomej ε. Podobnie jak w pierwszej części dowodu, wobec faktu, że na mocy tezy 1 wszystkie wyróżniki $\Delta^{*}_{a_{p_1+p_2+i}}$ dla $i = 0, \ldots, n-(p_1+p_2)$ traktowane jako nierówności kwadratowe na mocy tezy 1 mają ujemne Współczynniki przy drugiej potędze niewiadomej $a^{*}_{p_1+p_2+i}$, oznacza to, że wyjściowy układ równań (A.2) a dokładnie jego równoważna postać (A.7) ma dokładnie jedno rozwiązanie, co należało wykazać. Kończy to tym samym dowód poprawności i wykonalności kroku $p.4$. algorytmu SLS2S.

Dodatek B

Kod realizujący algorytm LC2S

Przedstawiony poniżej szkielet kodu implementującego algorytm LC2S realizuje jego wersję iteracyjną. W wyniku nie jest zwracany korzeń drzewa binarnego, powstaje natomiast plik `tree.log`, który zawiera tekstowy opis poszczególnych węzłów drzewa, z podaniem zależności między nimi.

Podając definicję najistotniejszych zmiennych tablicowych występujących w kodzie, podano wymiary tych tablic w odniesieniu do rozmiarów danych uczących (liczba wierszy, liczba atrybutów), z pełną świadomością, że taki sposób ich definiowania w dowolnym języku programowania byłby nieprawidłowy. Wyjątek ten uczyniono ze względu na czytelność kodu.

typ nazwa (opis)
`double size` liczba cech opisujących jeden przykład ze zbioru rozdzielanych danych.
`double lines` liczba przykładów w zbiorze rozdzielanych danych.
`double data[lines][size+2]` dwuwymiarowa tablica przechowująca rozdzielane dane (po zastosowaniu transformacji opisanej wzorem (3.1).
`int pointsAddedSoFar[size+1]` tablica, w której przechowywane są indeksy punktów przez, które mają przechodzić hiperpłaszczyzny rozdzielające.
`double q[size+1][size+2]` tablica przechowująca elementy ze zbioru $K \cup K_p^{(1)} \cup K_p^{(2)}$, przez które mają przechodzić hiperpłaszczyzny rozdzielające (określone w algorytmie LC2S w kroku $p.5$). Elementy tej tablicy są kopią odpowiednich wierszy tablicy `data`, ze względu na fakt, że w trakcie wyznaczania rozwiązania w kroku $p.5$ algorytmu LC2S ($p.4$ algorytmu SLS2S) ulegają one zmianie.
`double h[size+2]` tablica przechowująca aktualnie wyznaczone rozwiązanie $\boldsymbol{a}^*$ określające hiperpłaszczyzny $\boldsymbol{h}_1$ i $\boldsymbol{h}_2$.

typ nazwa (opis)
`double hMem[size+2]` tablica przechowujące kopię wyznaczonego na poziomie $p = 0$ rozwiązania. Odpowiednik $DB^{(0)}_{a^*}$ w kroku p.9 algorytmu LC2S.
`int flags[lines]` wykorzystanie tej tablicy umożliwiło zapis algorytmu w postaci iteracyjnej zamiast rekurencyjnej. Wartość elementu o indeksie i, oznacza, na którym poziomie p algorytmu LC2S ma być brany pod uwagę, wartość -1 oznacza, że punkt należy do zbioru Z (p.2 I p.17 algorytmu LC2S).
`int idClass1, idClass2` indeksy elementów tablicy `data`, należących do różnych klas, których odległość od siebie jest minimalna

Poniższa tabela zawiera zestawienie wybranych kroków algorytmu LC2S realizowanych przez poszczególne linie kodu źródłowego.

linie kodu	kroki algorytmu LC2S
11	wyznaczenie indeksów najbliżej położonych punktów należących do różnych klas.
14–24	p.3 i p.4
25–28	p.5 — implementacja kroku p.4 algorytmu SLS2S (zgodnie z podrozdziałem 4.2).
40–50	od p.15 przez p.10–p.22, realizacja powrotów rekurencyjnych, gdy $ZLE_p = \emptyset$.
71	p.10
95–130	Uzupełnianie pliku wynikowego `tree.log`.
110	p.20.
112, 124	p.21.

```
1  int LC2S(char *name, int parent,int treeNodeNr,int nextNode){
2      int idClass1, idClass2;
3      int h1Count=0,h2Count=0,h1tmp,h2tmp;
4      int timeToGo=0, doSolve=1,the0Level;
5      int addedP0, incorrect;
6      int size, lines;
7      readData(name,treeNodeNr,size,lines); //nazwa pliku z danymi: name+treeNode
8      int p=0; //level of algorithm LC2S;
9      for(int i=0;i<lines;i++)
10       flags[i]=p;
11     findTheClosestPair(idClass1,idClass2,size,lines); // w tablicy: data
12
```

```
13    while (!timeToGo){
14      if (doSolve){
15        if (h1Count==0){
16          pointsAddedSoFar[p]=idClass1;
17          h1tmp=1;
18        }
19        else h1tmp=0;
20        if (h2Count==0){
21          pointsAddedSoFar[p+h1tmp]=idClass2;
22          h2tmp=1;
23        }
24        else h2tmp=0;
25        for(i=0;i<h1Count+h2Count+h1tmp+h2tmp;i++)
26          for(int j=0;j<size+2;j++)
27            q[i][j]=data[pointsAddedSoFar[i]][j];
28        int epsilonZero=countSolution(size,h1Count+h1tmp+h2Count+h2tmp,q,h,p);
29          if (epsilonZero==1){
30            flags[addedP0]=-1;
31            p=0;
32            for(i=0;i<size+2;i++)
33              h[i]=hMem[i];
34            doSolve=0;
35            h1Count=h2Count=0;
36          }
37        }
38      }
39      doSolve=1;
40      do{
41        incorrect=0;
42        the0Level=(p==0 ? 1 :0);
43        if ((incorrect==0) && (p>0)){
44          if ((int)(data[pointsAddedSoFar[p-1]][size+1])==1)
45            h1Count--;
46        else
47            h2Count--;
48          p--;
49        }
50      }while ((incorrect==0) && (p>=0) && (!the0Level));
51      if (incorrect==0)
52        if (p==0)
53          timeToGo=1;
54        else{
55          if ((int)(data[pointsAddedSoFar[p-1]][size+1])==1)
56            h1Count--;
57          else
58            h2Count--;
59          p--;
60        }
61      else
```

```
62          if (p==size+1){
63            flags[addedP0]=-1;
64            p=0;
65            for(i=0;i<size+2;i++)
66              h[i]=hMem[i];
67            doSolve=0;
68            h1Count=h2Count=0;
69          }
70          else{
71            pointsAddedSoFar[p]=getSupportPoint(idClass1,idClass2,p,size,lines);
72            if (p==0){
73              addedP0=i;
74              for(int z=0;z<size+2;z++)
75                hMem[z]=h[z];
76            }
77            if (h1Count==size+1 || h2Count==size+1){
78              flags[addedP0]=-1;
79              p=0;
80              for(i=0;i<size+2;i++)
81                h[i]=hMem[i];
82              doSolve=0;
83              h1Count=h2Count=0;
84            }
85            else p++;
86          }
87      }
88      int is1=writeData(name,nextNode,size,lines,1,-1);   //nazwa pliku: name+nextNode
89                                          //    ^  ^ ( 1) dobre klasa 1,
90                                          //       ^ (-1) złe klasy 2
91      int is2=writeData(name,nextNode+1,size,lines,-1,1); //nazwa pliku: name+nextNode
92                                          //       ^ ^ (-1) zle klasa 1,
93                                          //         ^ ( 1) dobre klasy 2
94
95      FILE *tree;
96      if (treeNodeNr==parent){
97        tree = fopen("tree.log", "wt");
98        fprintf(tree,"ROOT (0): [");
99      }
100     else{
101       tree = fopen("tree.log", "at");
102       fprintf(tree,"parent: (%d)  node: (%d)  [",parent,treeNodeNr);
103     }
104     for(int z=0;z<size+2;z++)
105       fprintf(tree,"%8.15f ",h[z]);
106     fprintf(tree,"]\n");
107     fclose(tree);
108     int result=nextNode+2;
109     if (is1){
110       result=LC2S(name, treeNodeNr,nextNode,result);
```

```
111       if (is2)
112         result=LC2S(name, treeNodeNr,nextNode+1,result);
113       else{
114      tree = fopen("tree.log", "at");
115      fprintf(tree,"parent: (%d)  node: (%d)  klasa 2\n",treeNodeNr,nextNode+1);
116      fclose(tree);
117       }
118     }
119     else{
120       tree = fopen("tree.log", "at");
121       fprintf(tree,"parent: (%d)  node: (%d)  klasa 1\n",treeNodeNr,nextNode);
122       fclose(tree);
123       if (is2)
124         result=LC2S(name, treeNodeNr,nextNode+1,result);
125       else{
126         tree = fopen("tree.log", "at");
127         fprintf(tree,"parent: (%d)  node: (%d)  klasa 2\n",treeNodeNr,nextNode+1);
128         fclose(tree);
129       }
130     }
131     return result;
132 }
```

Bibliografia

[1] Słownik wyrazów obcych, Warszawa PWN, 1997

[2] *Encyklopedia Powszechna*,Warszawa, PWN, 1974

[3] Aitken A. C.: *Determinants and matrices*, Oliver and Boyd, 1951

[4] Basu M., Ho T. K.: The Learning Behavior of Single Neuron Classifiers on Linearly Separable or Nonseparable Input, *Proceedings of 1999 Int. 1 Joint Conference on Neural Networks*, Washington Dc, 1999

[5] Bishop C. M.: *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

[6] Camastra F.: Data dimensionality estimation methods: a survey, *Pattern Recognition*, vol. 36, nr 12, 2945–2954, 2003

[7] Cichosz P.: *Uczenie się maszyn*, Warszawa, PWN, 2000

[8] Cortes C., Vapnik V.: Support Vector Networks, *Machine Learning*, vol. 20, 273–297, 1995

[9] Cristianini N., Shawe-Taylor J.: *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000

[10] Cendrowska D.: Rekursywny algorytm optymalnego rozdzielania dwóch zbiorów, materiały konferencyjne *XII Konferencji Sieci i Systemy Informatyczne*, tom II,str. 417–424, Łódź, 2004

[11] Cendrowska D.: Strict Maximum Separability of Two Finite Sets, Algorithmic Approach, *International Journal of Applied Mathematics and Computer Science*, University of Zielona Góra Press, 2005

[12] Cendrowska D.: Hierarchical Binary Classifier, Algorithmic Approach, materiały konferencyjne *1st Polish and International PD Forum-Conference on Computer Science*, Łódź--Bronisławów, 2005

[13] Cendrowska D.: Hierarchiczny klasyfikator binarny wykorzystujący algorytm badania, optymalnej, liniowej rozdzielności dwóch zbiorów, materiały Międzynarodowych Warsztatów Doktoranckich, str. 169—174, Wisła, 2005

[14] Cendrowska D.: Hierarchical Classifier Based on Algorithm Finding Optimal Separability of Two Sets, materiały konferencyjne AI-METH, str. 35–36, Gliwice, 2005

[15] Daniecka D., Dubrawski A.: Attribute Selection for Neural Training of a Breast Cancer Diagnosis System, materiały konferencyjne: *the Third International Conference on Cognitive and Neural Systems*, Boston University, Boston, 1999

[16] Devijver P. A., Kittler J. V.: *Pattern Recognition. A Statistical Approach*, Prentice-Hall, 1982

[17] Duch W., J. Korbicz, L. Rutkowski, R. Tadeusiewicz (redaktorzy): *Sieci neuronowe*, Biocybernetyka i inżynieria biomedyczna, tom 6, Akademicka Oficyna Wydawniczna EXIT, Warszawa, 2000

[18] Duda R. O., Hart P. E., Stork D. G: *Pattern Classification*, Wiley Interscience, 1995

[19] Engelbrecht A. P.: *Computational Inteligence, An Introduction*, Wiley, Nowy Jork, 2002

[20] Franc V., Hlaváč V.: An iterative algorithm learning the maximal margin classifier, *Pattern Recognition*, nr 36, 1985–1996, 2003

[21] Jankowski N.: *Ontogeniczne sieci neuronowe*, Akademicka Oficyna Wydawnicza EXIT, 2003

[22] Gatnar E.: *Symboliczne metody klasyfikacji danych*, Warszawa, PWN, 1998

[23] Harel D.: *Rzecz o istocie informatyki, algorytmika*, Wydawnictwo Naukowo-Techniczne, 1987

[24] Ho Y.C., Kashyap R.L.: An Algorithm for Linear Inequalities and its Application, *IEEE Trans. on Electronic Computers*, 14, 1965, 683–688

[25] Jóźwik A.: A recursive method for the investigation of the linear separability of two sets, *Pattern Recognition*, vol. 16, no. 4, 429–431, 1983

[26] Jóźwik A.: Algorytm badania liniowej rozdzielności dwóch zbiorów i perspektywy jego wykorzystania do konstrukcji klasyfikatorów, *VI Konferencja Sieci i Systemy Informatyczne — teoria, projekty i wdrożenia*, materiały konferencyjne, 311–316, Łódź, 1998

[27] Jóźwik A.: Metoda badania liniowej rozdzielności dwu zbiorów skończonych w n-wymiarowej przestrzeni liniowej w zagadnieniach rozpoznawania obrazów *Prace Instytutu Organizacji i Kierowania, Cybernetyka stosowana i informatyka*, vol. 18, 1975

[28] Jóźwik A.: Nowy schemat testowania klasyfikatorów, *XII Konferencja Sieci i Systemy Informatyczne — teoria, projekty i wdrożenia*, materiały konferencyjne, 425–430, Łódź, 2004

[29] Konar A.: *Computational Inteligence*, Springer, Berlin, 2005

[30] Kozinec B. N.: Rekurentnyj algoritm razdelenia vypuklych obolochek dvuch mnozhestv (Recurrent algorithm separating convex hulls of two sets), *Algoritmy Obuchenia Raspoznavania (Learning Algorithms in Pattern Recognition)*, editor V. N. Vapnik, Sovetskoje radio, pp 43–50, Moskwa, 1973 (po rosyjsku)

[31] Lotlikar R., Kothari R.: Adaptive linear dimensionality reduction for classification, *Pattern Recognition*, vol. 33, nr 2, 185–194, 2000

[32] Mangasarian O. L.: Generalized Support Vector Machines, *Advances in Large Margin classifiers*, 135–146, MIT Press, 2000,
`ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps`

[33] McLachlan G. J.: *Discriminant Analysis and Statistical Pattern Recognition*, Wiley Interscience, Nowy Jork, 1992

[34] Mostowski A., Stark M.: *Algebra wyższa, część I*, Polskie Towarzystwo Matematyczne, Warszawa, 1953

[35] Nillson N. J.: *Learning Machines*, McGraw Hill, New York, 1965

[36] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes in C*, Cambridge University Press, 1992

[37] Prévost M.: Acceleration property for the E-algorithm and an aplication to the summation of series, `http://ano.univ-lille1.fr/pub/1993/ano295.ps`

[38] Russell S., Norvig P.: *Artificial Intelligence, A Modern Approach*, Prentice Hall, 2003

[39] Schürmann J.: *Pattern Classification*, Wiley-Interciene, 1996

[40] Swets D. L., Weng J.: Using Discriminant Eigenfeatures for Image Retrieval, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18 no. 8 str. 831–837, 1996

[41] Szczepanik P.S.: *Obliczenia inteligentne, szybkie przekształcenia i klasyfikatory*, EXIT, Warszawa 2004

[42] Vapnik V. N.: *The Nature of Statistical Learning Theory*, Springer, New York, 2000

[43] Turk M., Pentland A.: Eigenfaces for recognition, *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, str. 71–86, 1991

[44] Wang X., Paliwal K. K.: Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition, *Pattern Recognition*, vol. 36, nr 10, 2429–2439, 2003

[45] UCI Machine Learning Repository, `http://www.ics.uci.edu/~mlearn/MLSummary.html`