Uniwersytet Mikołaja Kopernika
w Toruniu

Wydział Fizyki, Astronomii
i Informatyki Stosowanej

Katedra Informatyki Stosowanej

Łukasz Itert

Komitety systemów klasyfikujących

Praca magisterska
pod kierunkiem prof. Włodzisława Ducha

Toruń, 2002

Bratu Szymonowi

Spis treści

Wstęp

Wśród wielu zadań stawianych Inteligencji Obliczeniowej (*Computational Intelligence*) klasyfikacja pełni rolę jedną z ważniejszych, jeśli nie najważniejszą. Nic dziwnego, skoro zagadnienie to coraz bardziej wkracza w nasze życie, a maszynowe systemy klasyfikujące spotkać można zarówno w placówkach medycznych, jak i w bankach. Stąd ciągłe zainteresowanie systemami klasyfikującymi oraz mnogość modeli usiłujących sprostać temu zadaniu. Pomimo ogromu różnych podejść i rozwiązań, stosowania rozmaitych technik obliczeniowych i specjalistycznych algorytmów, wielu badaczy w tej dziedzinie zdecydowało się jednak wytoczyć broń najcięższą – skorzystać z komitetów.

Komitety stanowią naturalną adaptację otaczającej nas rzeczywistości do rzeczywistości świata obliczeniowego. W społeczeństwie ludzkim często zdarza się, iż ważne decyzje nie są podejmowane jednoosobowo, lecz przez radę ekspertów z danej dziedziny. Podobnie jest z rozwiązywaniem trudnych problemów, co ma oczywiste i racjonalne uzasadnienie. Bywa jednak i tak, że nawet w gronie ekspertów decyzja musi zostać podjęta niedemokratycznie. Sytuacja taka ma miejsce wówczas, gdy badana tematyka jest tak rozległa, iż każdy ekspert specjalizuje się tylko w jej wąskim fragmencie. Wtedy trzeba zaufać jednej osobie – jej głos decyduje w największym stopniu o końcowej decyzji podejmowanej przez grono.

Zaprezentowaniu tematyki klasyfikacji, komitetów oraz zbadaniu wybranego, niedemokratycznego modelu komitetu poświęcona jest niniejsza praca. Model *CUC – Competent Undemocratic Committee* powstał w Katedrze Informatyki Stosowanej. Opracowanie to dotyczy jego implementacji numerycznej, przetestowania i analizy otrzymanych wyników.

Praca składa się z czterech głównych działów. Pierwszy rozdział zawiera podstawy zagadnienia klasyfikacji – wprowadza niezbędne pojęcia i definicje oraz porusza problematykę danych i modeli.

Drugi rozdział dotyczy systemów klasyfikujących – opisane zostają wybrane klasyfikatory używane w Inteligencji Obliczeniowej. Ta część pracy ma na celu zapoznanie z modelami użytymi do otrzymania danych numerycznych (a więc utworzenia samego komitetu). Omówiona zostaje idea każdego systemu, podstawowe założenia i zasada działania. Główna uwaga zostaje zwrócona na różnorodność stosowanych rozwiązań.

Rozdział trzeci to opis komitetów – wprowadza się podstawy dziedziny, przedstawia typową strukturę komitetu oraz sposoby działania najpopularniejszych z nich, a także kilka nowatorskich pomysłów.

W czwartym, ostatnim rozdziale znajduje się zasadnicza część pracy – model *CUC*. Zaprezentowane zostają idea modelu, jego numeryczne jądro oraz uzyskane wyniki dla popularnych zbiorów danych.

Wiele aspektów poruszanych w pracy jest przedstawionych w sposób niematematyczny (np. dział o danych można opisać bardzo sformalizowaną algebrą zbiorów), w całej rzeszy publikacji można znaleźć zupełnie inne podejścia. Celem przyświecającym takiej prezentacji poruszanej tematyki była chęć uczynienia pracy bardziej jasną, przejrzystą i wprowadzającą (również temu służą liczne przykłady), lecz także inną od pozostałych. Ponadto praca zawiera dużo rysunków, zgodnie z chińskim powiedzeniem „jeden obrazek wart jest więcej niż tysiąc słów". Warte komentarza są także występujące zwroty *angielskojęzyczne*. Ich pojawianie się jest czasami konieczne (ze względu na brak polskich odpowiedników), a czasami ma służyć ułatwieniu odnalezienia interesujących zagadnień w dominującej w tej dziedzinie literaturze angielskiej.

Wielkości wektorowe występujące w pracy oznaczono czcionką pogrubioną, wielkości macierzowe są dodatkowo podkreślone. Zbiory symbolizowane są przez czcionki kaligraficzne.

1 Zagadnienie klasyfikacji

1.1 Podstawowe pojęcia

Zagadnienie klasyfikacji jest problemem jasno sformułowanym. Należy stwierdzić przynależność danego przedmiotu do pewnej znanej grupy przedmiotów – klasy – stąd nazwa. Ustalenie klasy przedmiotu może być pomocne w zrozumieniu zarówno istniejących już danych, jak i nowych przypadków, które mogą się pojawić. Dlatego też klasyfikacja łączy w sobie dwie przeciwne strony osi czasu: na podstawie często wieloletnich badań tworzone są bazy danych z „ręcznie" sklasyfikowanymi przypadkami (umożliwiające w gruncie rzeczy powstawanie i sprawdzanie modeli), by później móc wybiec w przyszłość i próbować przewidywać klasy pojawiających się, nowych obiektów.

Maszynowa klasyfikacja powstała z dwóch powodów: po pierwsze po to, by odciążyć człowieka od ręcznej i wolnej pracy, po drugie, by móc analizować przypadki, których złożoności człowiek nie jest w stanie ogarnąć swoim umysłem. Mimo to człowiek z zagadnieniami klasyfikacji spotyka się niezwykle często i rozwiązuje ten problem najczęściej dość łatwo[1]. Każdy z nas, widząc np. nowy model samochodu i nie spotykając go wcześniej, potrafi stwierdzić, iż jest to właśnie samochód. Mózg nasz dokonuje zatem swoistej aproksymacji – na podstawie zgromadzonych danych (znanych lub widzianych wcześniej modeli samochodów) potrafi poprawnie sklasyfikować nieznany przypadek. Dzieje się to całkowicie nieświadomie, nikt z nas nie stara się przecież zapamiętać różnych modeli aut. Jednak klasyfikacja musi dokonywać się na przykładzie pewnych konkretnych cech – widząc samochód zauważamy karoserię, koła, czasami stwierdzamy ruch – cechy które z reguły bagatelizujemy, a to one wpływają na naszą decyzję. W przypadku zlecenia klasyfikacji technikom maszynowym problem staje się trudniejszy. To, co człowiek robi nieświadomie (widząc pędzące auto nie sprawdzamy czy ma koła!) – analiza cech przedmiotów – tu staje się głównym celem. Numeryczna klasyfikacja polega na takim nauczeniu systemu klasyfikującego, by mógł on – niczym mózg człowieka – poprawnie zaklasyfikować nie spotkany wcześniej obiekt do pewnej, znanej mu z procesu uczenia, grupy przedmiotów.

[1]Można wysunąć empiryczną hipotezę, że im łatwiej dane zagadnienie rozwiązuje człowiek, tym trudniej poradzą sobie z nim techniki maszynowe. Było tak ze słynnym problemem rozpoznawania płci osoby na podstawie jej zdjęcia twarzy.

Na podstawie powyższego wstępu wprowadzić można następujące, bardziej formalne, definicje, pojęcia oraz oznaczenia występujące w pracy:

cecha, atrybut – jest to najmniejsza „jednostka informacji" charakteryzująca dany obiekt, opisująca go. Cechy mogą być ciągłe (np. stężenie cukru we krwi) bądź też dyskretne (liczba potomstwa). Oznaczenie liczby cech – n.

klasa – jest to zbiór przedmiotów (obiektów) stanowiących pewną grupę, posiadających wspólne cechy. Klasami są np. samochody, drzewa, ludzie chorzy lub zdrowi. Oznaczenie liczby klas – K.

etykieta – jest to po prostu symboliczna nazwa klasy, często mająca postać skróconą (np. $C_1, C_2, C_3, \ldots$) lub numeryczną $(1, 2, 3, \ldots)$.

obiekt – przedmiot poddawany klasyfikacji lub też występujący w procesie uczenia, należący do pewnej klasy. Często z obiektami utożsamia się wektory, składowymi wektora są wówczas konkretne wartości cech obiektu. Przykładowo obiekt reprezentujący klasę „samochód" może mieć atrybuty: {`kolor, marka, rocznik`}, a przykładowy wektor cech składowe $\mathbf{X}$ ={`biały, opel, 1990`}.

przestrzeń cech – n-wymiarowa przestrzeń, w której obiekty opisywane są przez punkty wyznaczane przez wektory cech. Dla $n > 3$, bezpośrednie przeglądanie położeń wektorów jest utrudnione, stosuje się wówczas rzutowanie na płaszczyznę lub trójwymiarową przestrzeń wybranych cech. Przestrzeń cech służy także do graficznego prezentowania przebiegu granic decyzji.

granica decyzji – jest to granica wyznaczona przez hiperpowierzchnie (dla $n = 2$ są to linie proste, półproste, odcinki lub krzywe) oddzielające od siebie przedmioty należące do różnych klas. Granice decyzji są prostym i wygodnym narzędziem służącym do wizualizacji położenia klas w przestrzeni cech. Obszary wydzielone przez granice decyzji to tzw. **obszary decyzyjne**.

klasyfikator, system lub model klasyfikujący – numeryczny model dokonujący klasyfikacji.

klasa rzeczywista – prawdziwa klasa, do której należy obiekt.

klasa przewidziana, przewidywana – klasa, do której według klasyfikatora należy obiekt, może być różna od klasy prawdziwej.

Do rozwiązywania i analizowania procesu klasyfikacji potrzebne są zatem dwie rzeczy: dane oraz klasyfikatory. Zastaną one omówione w dwóch następnych podrozdziałach.

1.2 Dane do klasyfikacji

W Inteligencji Obliczeniowej klasyfikacja polega na zaprojektowaniu systemu klasyfikującego, a następnie na jak najlepszym nauczeniu go i przystosowaniu do rozpoznawania nowych przypadków. Uczenie polega zatem na analizie zbioru znanych obiektów, w zbiorze tym cechy, na podstawie których dokonywana będzie klasyfikacja, są *explicite* wyodrębnione i nazwane, podobnie jest z klasami. Niezbędne są dwa niezależne zbiory danych:

Zbiór treningowy – zbiór, na którym dany model się uczy, na podstawie którego adaptuje swoje parametry. Model ma pełny dostęp do danych treningowych, także do informacji o klasach wszystkich wektorów.

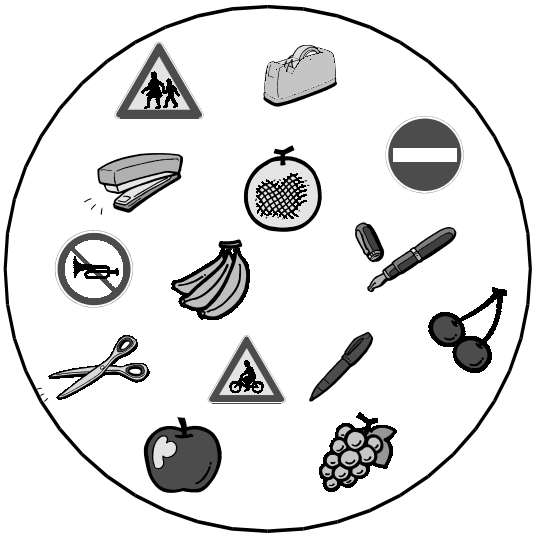
Zbiór testowy – zbiór służący do sprawdzania poprawności zaprojektowania i stopnia nauczenia modelu. Dostęp do klas wektorów jest zabroniony, ponieważ celem działania modelu jest właśnie przewidzenie klasy. Po procesie uczenia można porównać klasę przewidywaną z prawdziwą, czyli stwierdzić czy wystąpił błąd w klasyfikacji (badając tym samym dokładność klasyfikatora).

Zbiory treningowy i testowy powinny być ze sobą ściśle skorelowane, pod względem proporcji wektorów z poszczególnych klas, stopnia zaszumienia danych, wariancji poszczególnych cech, itp. Istnieją ogólnodostępne zbiory danych, dostępne w Internecie, dzięki którym można porównywać skuteczność różnych rozwiązań do omawianego problemu – czyli *de facto* porównać ze sobą różne modele. Często zdarza się jednak, że trzeba dane do klasyfikacji przygotować samemu, od podstaw – przykładowy zbiór takich danych zawiera rysunek 1.

Procedurę przygotowywania danych podzielić można na kilka etapów:

1. Należy wstępnie określić liczbę klas oraz poselekcjonować przedmioty na grupy, w obrębie których przedmioty wykazywały będą wzajemne podobieństwo. Ta część zadania zwykle należy do człowieka[2] i często

[2]Istnieją jednak metody służące do klasteryzacji tj. „samoistnego" podziału zbioru przedmiotów na grupy (klastry) obiektów wykazujących podobieństwo względem siebie. Więcej przeczytać można w [1, 14, 24, 33].



Rysunek 1: Przykładowy zbiór przedmiotów służący do przygotowania danych.

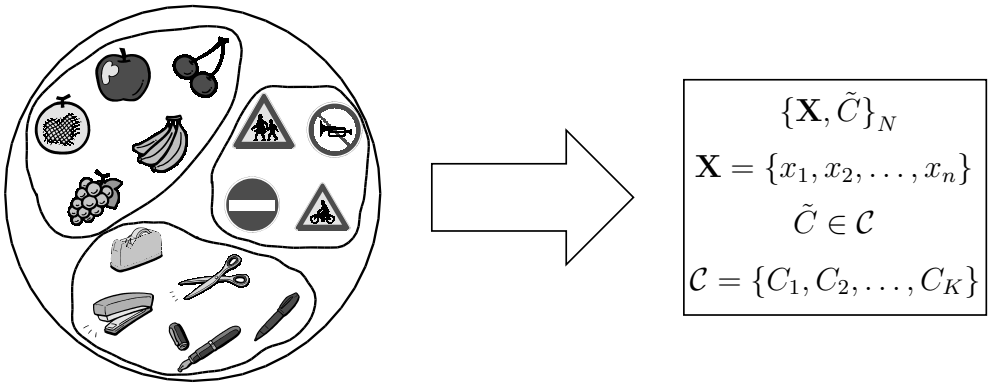
wymaga wiedzy eksperckiej (np. przy podziale danych medycznych na klasy o różnym stopniu ryzyka zapadalności na chorobę). Jest też ona szczególnie ważna, gdyż żaden system nauczony na złych danych nie będzie oczywiście poprawnie klasyfikował danych testowych.

2. Kolejnym etapem jest wyodrębnienie cech, czyli tych elementów danych, które będą miały największy wpływ na cały proces klasyfikacji. Wybór cech (w tym ich liczby) odgrywa bardzo dużą rolę, mało znaczące cechy nigdy nie zapewnią odpowiednich wyników, zbyt duża ich liczba powoduje nadmierną wielowymiarowość (a zatem złożoność) problemu, co ma bezpośredni wpływ na szybkość procesu uczenia i przetwarzania danych. O ważności wyboru cech świadczy też coraz większa liczba powstających systemów opierających swe działanie na tzw. ważeniu lub też selekcji cech[3].

3. Mając przygotowany zbiór główny, należy wyodrębnić w nim (zgodnie ze wspomnianymi założeniami) cześć treningową i testową. Pierwsza z nich powinna być większa od drugiej, co jest uzasadnione koniecznością zaprezentowania dużej liczby wzorców w procesie uczenia.

Całą omówioną procedurę przedstawia skrótowo rysunek 2.

[3]Najogólniej mówiąc, w systemach takich bada się, które cechy są najważniejsze w procesie klasyfikacji, a które prawie całkowicie zbędne. Szczegóły zaprezentowane są w [1].



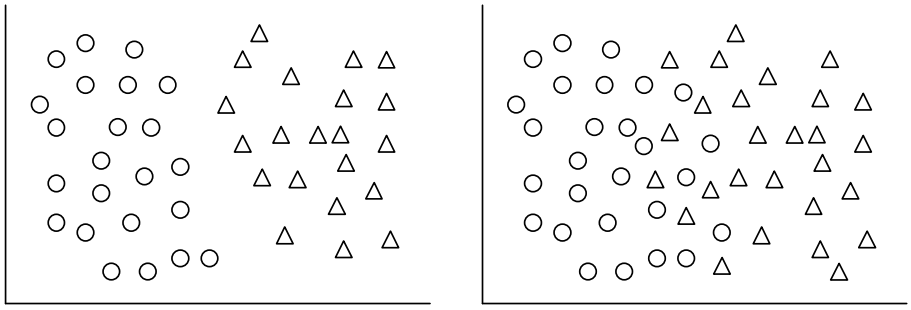
Rysunek 2: Przygotowywanie danych – wyodrębnianie (ekstrakcja) cech. Po podziale na K klas otrzymujemy N par wzorców uczących. Każdy wzorzec składa się z n-wymiarowego wektora cech oraz etykiety klasy – $\tilde{C}$.

Jeśli jednak nie można skorzystać ze zbioru testowego, stosuje się wówczas metody łączące uczenie i testowanie, a używające tylko jednego zbioru danych. Do najpopularniejszych należą:

1. Kroswalidacja (*crossvalidation* – *CV*). Zwykle mówi się o f-krotnej *CV*, w której pojedynczy zbiór zostaje podzielony na f części. Przeprowadzając f-krotny cykl obliczeń, każdorazowo wyłączamy inny zbiór ($\frac{1}{f}$-tą zbioru głównego) z cyklu treningowego i dokonujemy na niej oceny nauczenia systemu (czyli testu). Uśrednione wyniki f testów stanowią końcowy wynik testowy. Istnieje kilka odmian *CV*, różniącym się sposobem podziału zbioru głównego. Najważniejszą z nich jest *stratified CV*, w której w każdym zbiorze zostają zachowane proporcje w liczbie wektorów z poszczególnych klas.

2. *Leave One Out* (*L1O*) – jest to specyficzna odmiana *CV*, dla której f jest równe liczebności zbioru wektorów. W metodzie tej każdorazowo ze zbioru wyłączamy jeden wektor, który po zakończeniu procesu uczenia na pozostałych danych, posłuży do przeprowadzenia testu. Dodatkowo metodę tę stosuje się często, gdy chcemy poznać szczegółowe informacje o każdym pojedynczym wektorze ze zbioru danych.

3. *Bootstrapping* – w najprostszej odmianie tej metody, zamiast cyklicznie wykonywanych testów na podzbiorze danych o ustalonej wielkości (*CV*), dokonujemy analizy na zbiorze złożonym z wektorów, które zostały losowo (i z powtórzeniami) wybrane do zbioru testowego. Uczenie systemu odbywa się na niewytypowanych wektorach.



Rysunek 3: Przykład dwuklasowych zbiorów danych. Z lewej strony zbiór klas rozłącznych, z prawej obserwujemy nakładanie się klas.

Sytuacja przedstawiona na rysunku 2, gdy mamy do czynienia z klasami rozłącznymi, zdarza się niezwykle rzadko. Zdecydowanie częstszym zjawiskiem jest wzajemne nakładanie się klas (rysunek 3), przynależność obiektu do każdej klasy opisywana jest wówczas rozkładem prawdopodobieństwa.

Zebrane dane nie zawsze mogą być użyte w sposób bezpośredni przez klasyfikator (np. z powodu dużych rozbieżności wartości cech, innych skal wielkości, itp.), wówczas powinny zostać one wstępnie przetworzone. Taki zabieg często poprawia zdolność klasyfikacji danego modelu, choć nie jest to regułą i konieczność transformacji najlepiej zweryfikować doświadczalnie. Do najpopularniejszych metod wstępnej obróbki należą:

1. **Normalizacja**

   Najczęściej normalizacja wektora kojarzy się z takim przeskalowaniem jego składowych, by długość nowo powstałego wektora wynosiła $\|1\|$[4]:

   $$x'_k = \frac{x_k}{\sum_{f=1}^{n} x_f}.$$

   W uczeniu maszynowym normalizacja danych polega jednak na takim przeskalowaniu każdego wektora cech reprezentującego pojedynczy wzorzec uczący, by po transformacji cechy wektora zawierały się w przedziale [0,1]. Nowy wektor $\mathbf{X}'$ tworzymy zgodnie z wzorem:

   $$x'_k = \frac{x_k - x_k^{min}}{x_k^{max} - x_k^{min}},$$

[4] $\|\|$ oznacza metrykę euklidesową.

gdzie x_k są składowymi wektora $\mathbf{X}$ przed normalizacją, x_k^{max} jest wartością maksymalną jego k-tej cechy (z całego zbioru), natomiast x_k^{min} jej wartością minimalną.

2. **Standaryzacja**

   Zmienną losową nazywamy standaryzowaną, gdy jej wartość przeciętna wynosi 0 a wariancja 1.
   W przeciwieństwie do normalizacji, która do przeskalowania wektora nie używa informacji o rozkładzie jego cech, standaryzacja dokonuje transformacji zmiennych biorąc pod uwagę ich wariancję, co zapewnia proporcjonalne przeskalowanie w przypadku wartości cech znacznie odbiegających od średniej, a pojawiających się sporadycznie i mogących być wynikiem przypadkowych czynników, np. błędami pomiaru. Współrzędne wektora transformujemy według wzoru:

$$x'_k = \frac{x_k - \bar{x}_k}{\sigma(x_k)},$$

   gdzie $\sigma(x_k)$ jest odchyleniem standardowym k-tej cechy:

$$\sigma(x_k) = \frac{1}{N-1}\sum_{i=1}^{N}(x_k^i - \bar{x}_k)^2,$$

   a $\bar{x}_k$ średnią wartością cechy z całego zbioru:

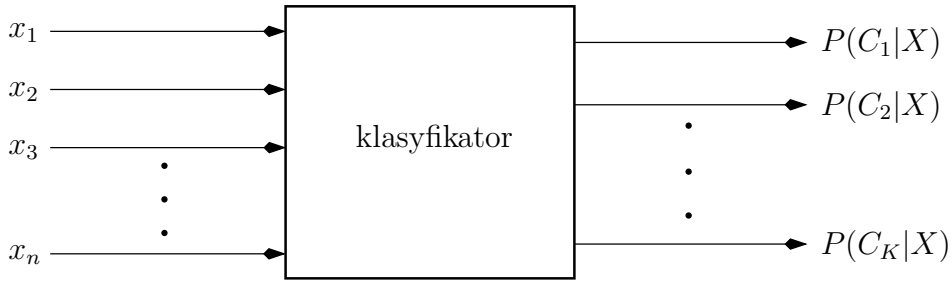
$$\bar{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_k^i.$$

Normalizację lub standaryzację należy wykonywać jednocześnie na zbiorze treningowym i testowym. Dane, które nie są w żaden sposób wstępnie przetwarzane, określane są terminem *raw data*.

1.3 System klasyfikujący

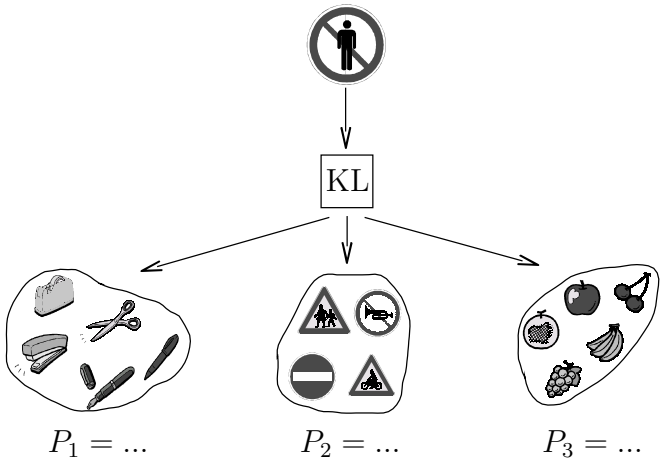
Przyjrzyjmy się ogólnej budowie, działaniu i właściwościom klasyfikatora, który w tym rozdziale traktowany będzie jako „czarna skrzynka", bez wnikania w jego wewnętrzną strukturę. Schemat takiego klasyfikatora przedstawia rysunek 4.

Zadanie klasyfikatora to transformacja danych wejściowych do danych wyjściowych. Klasyfikator posiada n wejść oraz K wyjść, co jest uwarunkowane

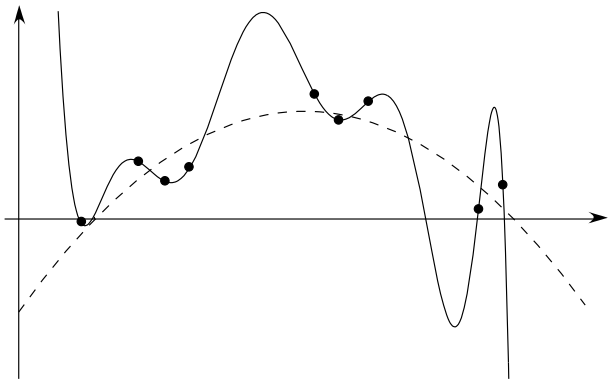


Rysunek 4: Ogólny model klasyfikatora. Wektor danych wejściowych $\mathbf{X}$ zostaje zanalizowany przez klasyfikator. Otrzymujemy serię prawdopodobieństw przynależności do poszczególnych klas.

wymiarem wzorca uczącego oraz liczbą klas. Po analizie otrzymujemy serię prawdopodobieństw przynależności analizowanego wektora do każdej klasy (rysunek 5), idealna sytuacja występuje, gdy jedno z prawdopodobieństw wynosi 1, a pozostałe 0, jednak tak zdarza się niezwykle rzadko (w przypadku klas rozłącznych). Z punktu widzenia matematycznego otrzymane wyniki są prawdopodobieństwami warunkowymi: $P(A|B)$ oznacza prawdopodobieństwo zdarzenia A pod warunkiem, że miało miejsce zdarzenie B. Analogicznie zapis $P(C|\mathbf{X})$ oznacza wystąpienie klasy C pod warunkiem, że pojawił się wektor $\mathbf{X}$. Statystyka Bayesowska określa takie prawdopodobieństwo mianem *a posteriori* – oznacza to, iż otrzymaliśmy je w wyniku analizy zbioru danych.



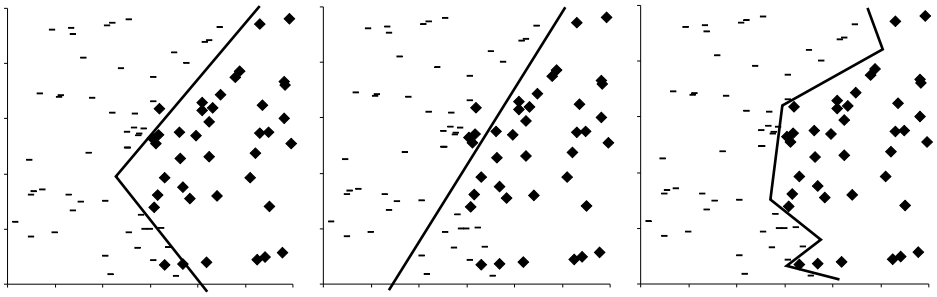
Rysunek 5: Klasyfikacja nieznanego przypadku. Zadanie klasyfikatora (KL) to określenie prawdopodobieństw P_1, P_2, P_3.



Rysunek 6: *Overfitting* występujący w klasycznej interpolacji. Wielomian 10-tego stopnia idealnie przechodzi przez zadane punkty, dane pochodzą jednak z zaszumionej funkcji kwadratowej.

Ze względu na sposób wykorzystywania wiedzy zawartej w przykładach zbioru treningowego, rozróżniamy klasyfikatory uczące oraz nie uczące się. Uczenie się z danych może przybierać różne formy: może zmieniać się struktura całego układu (liczba i struktura połączeń neuronów w sieciach, węzłów w drzewach) bądź też tylko parametry adaptacyjne (np. wagi połączeń pomiędzy neuronami). Wszystko to ma zapewnić optymalne dobre dopasowanie się do danych. Również w samym uczeniu się można wyróżnić różne kategorie, najpopularniejszy podział to uczenie nadzorowane, z nauczycielem (*supervised learning*) i nienadzorowane, bez nauczyciela (*unsupervised*). W uczeniu nadzorowanym zawsze istnieje pojęcie zewnętrznego „nauczyciela", który zna odpowiedź na postawione pytanie i naprowadza na nią ucznia. Uczenie bez nauczyciela to uczenie spontaniczne, polegające na samoistnym odkrywaniu reguł i nieznanych zależności zgodnie z założoną filozofią działania.

Ważnym pojęciem w procesie klasyfikacji jest **generalizacja**. Celem procesu uczenia systemu nie jest takie nauczenie, by osiągać jak najlepsze wyniki na zbiorze treningowym (gdy przeprowadzimy na nim test), ale by osiągać maksymalnie dobre wyniki na części testowej. Generalizacja jest to więc zdolność systemu do otrzymywania poprawnych wyników dla danych nigdy wcześniej nie prezentowanych, czyli nie biorących udziału w procesie uczenia.
Optymalny stopień nauczenia znajduje się pomiędzy dwoma granicznymi pojęciami: niedouczeniem (*underfitting*) i przeuczeniem (*overfitting*). Klasyfikator pełni rolę interpolatora (lub aproksymatora), szuka, bazując na części treningowej, nieznanej zależności pomiędzy danymi wejściowymi (cechy obiektów) a wyjściowymi (klasy, do której obiekty należą). Uwidacznia się



Rysunek 7: Od lewej: optymalna granica decyzji oraz granice wynikające z niedouczenia i przeuczenia.

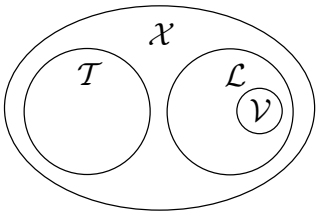
tu analogia do klasycznej interpolacji znanej z analizy numerycznej, również tam dobranie nieodpowiedniego modelu prowadzić może do kuriozalnych wyników (rysunek 6). Informacja „ukryta" w danych zawiera sygnał właściwy oraz pewien szum (związany np. z niepewnością pomiaru) – przekłada się to na wariancję zbioru oraz w wielu wypadkach powoduje niemożliwość wyznaczenia optymalnej granicy decyzyjnej. Jeśli występuje niedouczenie, klasyfikator może nie wykryć właściwego sygnału, jeśli zaś przeuczenie, może nastąpić zbytnie dopasowanie szukanej zależności do szumu. Zjawisko niedouczenia i przeuczenia klasyfikatora przedstawia rysunek 7.

Istnieje kilka metod poprawiających generalizację, stosowanych w zależności od rozpatrywanego modelu, najpopularniejsze z nich to:

- Użycie odpowiednio dużego zbioru treningowego – zwiększa to szansę na wykrycie właściwych zależności pomiędzy wejściem i wyjściem klasyfikatora oraz uwidacznia granicę stosunku sygnał/szum. Istnieje szereg heurystycznych reguł odnośnie liczby wzorców uczących w zależności od uczonego klasyfikatora, np. reguła mówiąca, iż dla sieci neuronowej liczba ta powinna być przynajmniej 30 razy większa od liczby wszystkich połączeń pomiędzy neuronami [31].

- Odpowiedni dobór parametrów układu, gdy istnieje możliwość startu z różnymi wartościami początkowymi (wagi sieci neuronowych) lub różną kolejnością przetwarzania argumentów (drzewa decyzji), należy zbadać możliwie szerokie spektrum zmienności parametrów.

- Stosowanie członów regularyzacyjnych oraz dodawanie do danych szumu – zabiegi te mają wygładzić granice decyzji. W większości przypadków funkcje, których odwzorowań uczą się klasyfikatory, są gładkie



Rysunek 8: Podział zbioru danych $\mathcal{X}$ na część testową ($\mathcal{T}$) i treningową ($\mathcal{L}$). Dodatkowo w części treningowej wydzielono zbiór walidacyjny ($\mathcal{V}$).

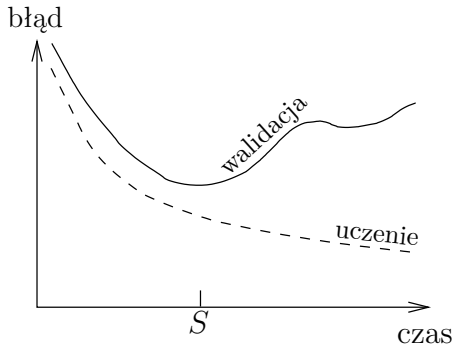
i nieliniowe.

- Często szkodliwym czynnikiem jest też zbyt długie uczenie.

Metody poprawy generalizacji są w większości poprawkami doświadczalnymi, wynikającymi z eksperymentów i prowadzonych obserwacji, choć istnieją także próby ich teoretycznego uzasadnienia[5].

Proces uczenia polega na cyklicznym analizowaniu zbioru danych tak, by błąd klasyfikacji na zbiorze testowym był możliwie najmniejszy. W związku z tym często wprowadza się pojęcie dodatkowego zbioru, tzw. zbioru walidacyjnego (rysunek 8). Zbiór walidacyjny pełni rolę zbioru testowego biorącego udział bezpośrednio w fazie uczenia (główny zbiór testowy służy do badania klasyfikacji już po zakończeniu nauki) a służy do sprawdzenia stopnia nauczenia systemu (proces sprawdzania stopnia nauczenia nazywamy walidacją). Jest to ważne w przypadku modeli posiadających dużą liczbę adaptacyjnych parametrów – w trakcie uczenia należy co pewien czas sprawdzać, czy zestaw parametrów prowadzi do pożądanych wyników. Jeśli nie, można powtórzyć obliczenia z innymi założeniami lub wartościami początkowymi. By zatem zmaksymalizować generalizację jednocześnie zachowując uczciwość i nie używać składników zbioru testowego w procesie uczenia (tzn. nie uczyć systemu tak, by wypadł najlepiej na zbiorze testowym), należy skorzystać ze specjalnie wyodrębnionego zbioru. Ważnym aspektem jest także czas uczenia. Optymalnym punktem przerwania uczenia jest moment, gdy pomimo ciągłego spadku wartości błędu na zbiorze treningowym, błąd zbioru walidacyjnego zaczyna rosnąć (następuje przeuczenie i zbytnie dopasowanie się do danych uczących), co ilustruje rysunek 9.

[5]Np. metoda MDL – Minimum Description Length [6].



Rysunek 9: Błąd klasyfikacji w funkcji czasu uczenia. Punkt S wyznacza optymalny moment przerwania nauki.

1.4 Jakość i dokładność klasyfikacji

Połączenie danych i modeli prowadzi do powstania wyników, których jakość oraz wiarygodność należy umieć ocenić i zinterpretować. W tym celu powstały metody oceny dokładności[6] oraz jakości klasyfikacji, niezależnie od rodzaju modelu z jakim mamy do czynienia. Pierwszym i zarazem największym narzędziem służącym do oceny wyników jest macierz konfuzji.

Macierz konfuzji (*confusion matrix*) przedstawia wyniki klasyfikacji poprzez podanie liczby poprawnie klasyfikowanych wektorów we wszystkich klasach. Dodatkowo forma macierzy umożliwia obliczenie błędnie klasyfikowanych wektorów, w tym, ile wektorów zostało błędnie przypisanych do której klasy. Mamy więc jasny obraz z którą klasą dany model radzi sobie lepiej, a z którą gorzej. Kolumny macierzy reprezentują klasę prawdziwą, wiersze klasę przewidzianą – suma poszczególnych kolumn musi być równa liczbie wektorów w danej klasie. Jeżeli interesują nas tylko poprawne klasyfikacje powinniśmy skupić uwagę na wartościach diagonalnych.

Macierz konfuzji może mieć dwie zasadnicze formy: przedstawiać liczbę lub procent (z danej klasy) poprawnie sklasyfikowanych wektorów.

Przykład
Rozważmy trzyklasowy (C_a, C_b, C_c) zbiór wektorów o liczebności wektorów w poszczególnych klasach wynoszących odpowiednio $20, 30, 40$. Niech macierz konfuzji przedstawiona będzie w tabeli 1, odpowiadająca jej macierz

[6]Często wprowadza się nieformalną definicję, iż dokładny klasyfikator to taki, którego błąd jest mniejszy, niż błąd popełniany podczas przypadkowego zgadywania klasy.

Tabela 1: Macierz konfuzji z przykładu 1 zawierająca dane o liczbie wektorów.

	C_a	C_b	C_c
C_a	15	2	5
C_b	4	25	3
C_c	1	3	32

Tabela 2: Procentowa macierz konfuzji dla przykładu 1. Dane są procentowym odpowiednikiem wyników z tabeli 1.

	C_a	C_b	C_c
C_a	75%	7%	12%
C_b	20%	83%	8%
C_c	5%	10%	80%

procentowa zawarta jest w tabeli 2. Z powyższych tabel wynika, że w klasie C_a 75% (15 z 20-stu) wektorów jest klasyfikowanych poprawnie, natomiast 5 wektorów zostało przypisanych do złych klas: 4 do klasy C_b i 1 do klasy C_c. Podobne informacje można wydobyć dla innych klas. Warto wspomnieć, że dla idealnego klasyfikatora macierz konfuzji ma postać przedstawioną w tabeli 3.

Tabela 3: Macierz konfuzji dla bezbłędnego klasyfikatora.

	C_a	C_b	C_c
C_a	100%	0%	0%
C_b	0%	100%	0%
C_c	0%	0%	100%

Jeżeli dokładność klasyfikacji nie jest wystarczająca, można spróbować następującego zabiegu. Oznaczamy niektóre przypadki (wybór ich może być dokonany nawet ręcznie – na podstawie obserwacji położenia w przestrzeni cech) jako niedające się sklasyfikować, tj. przypisujemy je do specjalnie utworzonej, dodatkowej klasy C_r. Załóżmy, że zagadnienie jest dwuklasowe (klasy: C_+ (N_+ wektorów) oraz C_- (N_-)), ponadto znana jest dla niego macierz konfuzji:

	C_+	C_-
C_+	N_{++}	N_{-+}
C_-	N_{+-}	N_{--}
C_r	N_{+r}	N_{-r}
	N_+	N_-

gdzie N_{+r} i N_{-r} są odpowiednio liczbami wektorów z klasy C_+ i C_- przypisanych do klasy C_r.
Dzieląc wszystkie wielkości przez $N = N_+ + N_-$ (liczba wektorów) można bazować na prawdopodobieństwach wyrażonych względem całego zbioru:

	C_+	C_-
C_+	P_{++}	P_{-+}
C_-	P_{+-}	P_{--}
C_r	P_{+r}	P_{-r}
	P_+	P_-

Prawdopodobieństwa aprioryczne zapisać możemy jako:

$$P_+ = P(C_+) = N_+/N,$$

$$P_- = P(C_-) = N_-/N$$

lub:

$$P_+ = P_{++} + P_{+-} + P_{+r}\ , \quad P_- = P_{-+} + P_{--} + P_{-r},$$

$$P_+ + P_- = 1.$$

Klasyfikację zbioru testowego zwiększyć można odrzucając pewne przypadki ze zbioru treningowego i wprowadzając wartości graniczne (progi), wyznaczane poprzez minimalizację odpowiedniej funkcji kosztu, których nieprzekroczenie spowoduje umieszczenie wektora w zbiorze o klasie C_r [12].

Dodatkowo dla zagadnienia dwuklasowego wprowadzić można następujące pojęcia:

- Całkowita dokładność (*overall accuracy*):

  $$A = P_{++} + P_{--}$$

  jest liczbą poprawnie sklasyfikowanych wektorów (z obu klas).

- Całkowity błąd (*overall error rate*):

$$L = P_{-+} + P_{+-}$$

liczba wektorów błędnie sklasyfikowanych.

- *Overall rejection rate*:

$$R = P_{+r} + P_{-r} = 1 - L - A$$

całkowita liczba wektorów niesklasyfikowanych (odrzuconych).

- Czułość (*sensitivity*):

$$S_+ = P_{+|+} = P_{++}/P_+$$

opisuje zdolność wykrywania wszystkich osobników rzeczywiście należących do danej klasy. Jest wyrażana przez prawdopodobieństwo warunkowe przypisania wektora do klasy C_+ pod warunkiem, że wektor jest rzeczywiście z tej klasy, tj. $P_{+|+} = P(C_+|\mathbf{X} \in C_+)$.

- Specyficzność (*specificity*):

$$S_- = P_{-|-} = P_{--}/P_-$$

jest odpowiednikiem czułości dla klasy C_-.

Korzystając z powyższych definicji można zauważyć wiele interesujących zależności, np.:

$$A = P_+S_+ + P_-S_-,$$

o których więcej można poczytać w [12].

2 Wybrane systemy klasyfikujące

W niniejszym rozdziale przedstawione zostaną wybrane systemy klasyfikujące. Nie jest to rozdział przeglądowy, a wybór omówionych modeli nie jest dziełem przypadku. Opisane zostaną te systemy, z których możliwe było uzyskanie danych numerycznych do stworzenia komitetu tj. powstałe w Katedrze Informatyki Stosowanej i zaimplementowane numerycznie: sieci neuronowe *FSM*, *IncNet*, drzewo *SSV* oraz klasyfikator kNN. Ponieważ większość użytych klasyfikatorów stanowi jednak rozbudowane lub przekształcone wersje modeli bazowych istniejących wcześniej, ważnym wydaje się być zaprezentowanie ich wzorców i koncepcji początkowych.

Porównanie klasyfikatorów ma na celu zaprezentowanie ich różnorodności zarówno w budowie, idei, na której się opierają, jak i działaniu – stąd pojawienie się podrozdziałów.

Początki klasyfikacji to metody statystyczne, których podstawy matematyczne powstały nawet kilkaset lat temu. Metody statystyczne opierają swoje działanie na zupełnie innej filozofii niż modele maszynowe, wzmianka o nich ma jedynie na celu ukazanie kontrastu pomiędzy nimi i dalszymi klasyfikatorami. Statystyka posługuje się wielkościami średnimi, odchyleniami, wariancjami oraz rozkładami prawdopodobieństwa. Jednak we współczesnej klasyfikacji w znakomitej większości przypadków żadne informacje na temat rozkładów prawdopodobieństwa danych nie są znane – jedyną informację czerpiemy ze zbioru oznaczonych przykładów. Właśnie taki sposób podania wiedzy potrafią wykorzystać metody *machine learning*.

2.1 Sieci neuronowe

Sieci neuronowe (*NN – Neural Networks*) są bez wątpienia najpopularniejszymi i najczęściej używanymi klasyfikatorami występującymi w Inteligencji Obliczeniowej. Budowa i działanie sieci wzorowane są na naturalnych sieciach neuronowych, w tym na największej – ludzkim mózgu. Sieci składają się z elementarnych komórek zwanych neuronami, których rozmieszczenie, połączenie, nadane właściwości oraz sposób wzajemnej współpracy i wspólnego uczenia mają całościowy wpływ na końcowy efekt działania sieci. Jednym z powodów ogromnej popularności NN jest zapewne bogactwo parametrów którymi dysponują konstruktorzy sieci (od struktury sieci, poprzez różne modele neuronów, aż do sposobu uczenia). Nie bez znaczenia są też pojawiające

Rysunek 10: Ogólny model neuronu. Neuron charakteryzowany jest przez p wejść i jedno wyjście (y), jego wnętrze może przybierać różne formy.

się odkrycia neurobiologiczne, które szybko otrzymują numeryczną postać[7].

Neurony. Podstawowymi elementami sieci są neurony – jednostki przetwarzające informacje – zaprojektowane na wzór neuronu biologicznego. Neurony charakteryzowane są głównie przez funkcję aktywacji – opisującą stan pobudzenia neuronu w zależności od pojawiających się bodźców wejściowych. Istnieje wiele odmian neuronów, różniących się między sobą, jedna rzecz pozostaje wszak niezmiennicza – neuron zawsze posiada wiele wejść i jedno wyjście (rysunek 10).

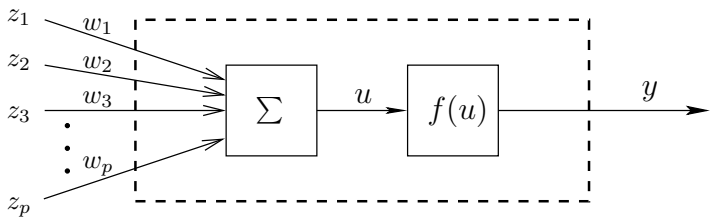
Budowa i działanie sieci. Połączenie neuronów w struktury definiuje pojęcie warstwy. Sieci neuronowe posiadają budowę warstwową, każda warstwa ma w sieci ściśle określone zadanie. W typowej sieci neuronowej wyróżnić można następujące warstwy i określić pełniące przez nie role:

- Warstwa wejściowa – warstwa, poprzez którą do sieci dostarczane są dane – nie występuje w niej żadne przetwarzanie informacji. W zagadnieniu klasyfikacji liczba neuronów w tej warstwie jest równa wymiarowi wektora cech.
- Warstwa lub warstwy ukryte – główna część sieci odpowiadająca za rodzaj odwzorowania reprezentowanego przez sieć.
- Warstwa wyjściowa – otrzymujemy z niej wynik odwzorowania – dla klasyfikacji liczba wyjść równa jest liczbie klas.

Działanie sieci polega na poszukiwaniu nieznanego odwzorowania Y od danych wejściowych ($\mathcal{X}$) do wyjściowych ($\mathcal{Y}$):

$$Y = f(\mathcal{X}) \,, \qquad Y : \mathcal{X} \to \mathcal{Y}.$$

[7]Tak było np. z regułą Hebba [27, 33].



Rysunek 11: Model sztucznego neuronu dla sieci *MLP*. Składowe wejściowego sygnału **Z** są mnożone przez odpowiadające im wagi. Wartość wyjściowa neuronu jest funkcją sumy ważonej u.

To, jaki rodzaj odwzorowanie sieć będzie mogła realizować, zależy od dwóch czynników: liczby warstw sieci (także od liczby neuronów w każdej warstwie) oraz funkcji aktywacji neuronów.

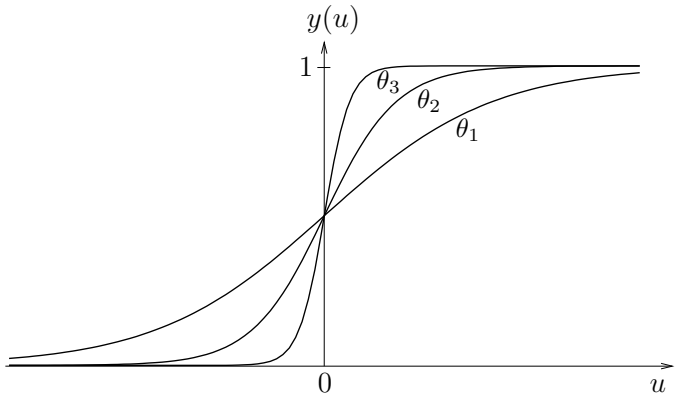
Początkowe sieci neuronowe (perceptrony) potrafiły rozwiązywać tylko problemy liniowo separowalne – inaczej mówiąc granice decyzji takich sieci były tylko hiperpłaszczyznami. Dalszy rozwój dziedziny usunął te ograniczenia. Współczesne badania nad sieciami neuronowymi koncentrują się nad tworzeniem modeli umożliwiających głównie nieliniowe odwzorowania oraz na wprowadzaniu coraz to wydajniejszych algorytmów uczenia.

Uczenie. Uczenie jest w sieciach procesem najbardziej zasygnalizowanym spośród wszystkich klasyfikatorów. Proces ten polega na ciągłej adaptacji parametrów – wag łączących neurony, ważnym zagadnieniem jest także dobór wag początkowych. Wprowadzenie ciągłych (w odróżnieniu od funkcji progowej) funkcji aktywacji umożliwiło zastosowanie wielu gradientowych metod uczenia, których rozmaite modyfikacje stanowią, pomimo prób wprowadzania np. algorytmów genetycznych, do dziś podstawę uczenia.

Klasyfikacja. Sieć o K wyjściach może rozpoznawać K różnych typów obiektów (klas). O przynależności do danej klasy decyduje najwyższa wartość wyjściowa danego neuronu.

2.1.1 Sieć MLP

Wielowarstwowe sieci neuronowe (*MLP – **M**ulti**L**ayer **P**erceptron)* są najpopularniejszą rodziną sieci neuronowych. Stanowią one uniwersalne narzędzie obliczeniowe, którego jednym z zastosowań jest klasyfikacja danych. Podstawowe cechy sieci *MLP*, oprócz neuronu przedstawionego na rysunku 11, to:



Rysunek 12: Sigmoidalna funkcja aktywacji (1) dla różnych wartości parametru θ ($\theta_1 < \theta_2 < \theta_3$).

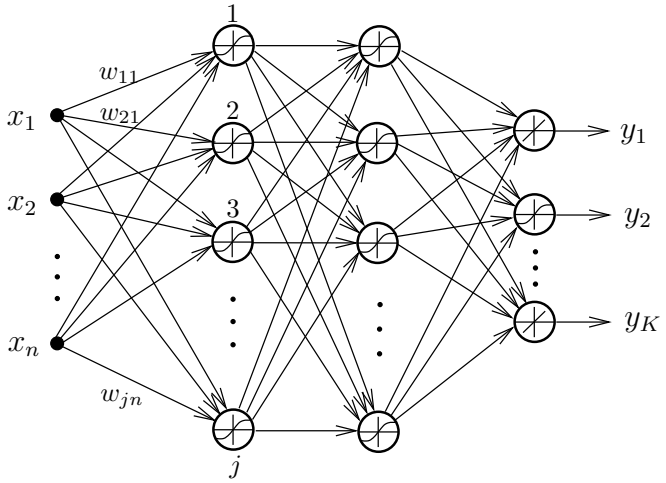
jednokierunkowy przepływ sygnału przez sieć, przynajmniej jedna warstwa ukryta oraz sigmoidalne funkcje aktywacji neuronów (rysunek 12):

$$y(u) = \frac{1}{1 + \exp(-\theta u)}, \tag{1}$$

gdzie u jest sumą ważoną sygnałów wejściowych neuronu:

$$u = \sum_{s=1}^{p} w_s z_s. \tag{2}$$

Pierwsze sieci jednokierunkowe miały tylko jedną warstwę ukrytą oraz neurony z liniowymi funkcjami aktywacji. Globalne działanie takiego układu mogło być tylko liniowe. Kolejnym krokiem było więc dodanie następnej warstwy ukrytej, jednak nie mogły znajdować się w niej neurony liniowe (wielowarstwowa sieć liniowa jest *de facto* równoważna sieci liniowej jednowarstwowej, gdyż kombinacja liniowa funkcji liniowych jest funkcją liniową). Konieczne stało się użycie funkcji nieliniowych we wszystkich warstwach ukrytych. Działanie takiej sieci było przez wiele lat bolączką badaczy poszukujących skutecznego algorytmu uczenia. Z upływem czasu, po ostatecznym pokonaniu wszystkich przeszkód, typowa sieć *MLP* zyskała strukturę przedstawioną na rysunku 13. Niestety, nie ma żadnych bezpośrednich wskazówek na temat dostosowania budowy sieci do analizy konkretnego problemu. Wiadomo, że dla zagadnień klasyfikacji liczba węzłów wejściowych powinna być równa liczbie cech danych, liczba neuronów wyjściowych natomiast równa liczbie klas. Ponieważ działanie sieci *MLP* polega na podziale przestrzeni danych wejściowych na podprzestrzenie (rysunek 14), można założyć, mając wstępne



Rysunek 13: Sieć *MLP* o dwóch warstwach ukrytych. Sieć posiada n wejść, dwie warstwy ukryte (w pierwszej występuje j neuronów) z sigmoidalnymi funkcji aktywacji oraz warstwę wyjściową (K-elementową). Neurony w ostatniej warstwie mogą być zarówno sigmoidalne, jak i liniowe.

informacje o rodzaju problemu, ilu warstw ukrytych należy użyć (z [36] wiadomo, że trójwarstwowa sieć *MLP* może rozwiązywać dowolne problemy nieliniowe), nie ma natomiast żadnych deterministycznych wskazówek na temat pożądanej liczby neuronów w warstwach ukrytych.

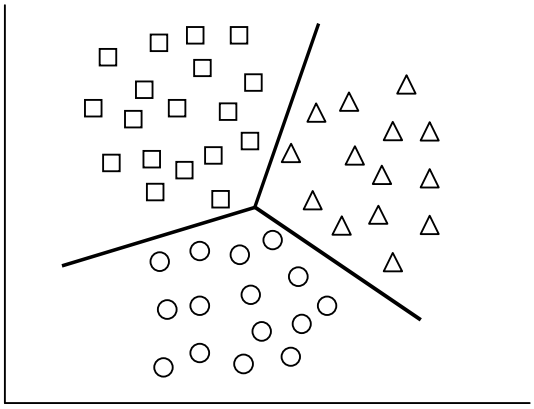
Uczenie sieci *MLP* przebiega zazwyczaj z nauczycielem (dostarczającym sygnał $\tilde{y}$) i polega na minimalizacji błędu średniokwadratowego:

$$E = \frac{1}{2}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \,, \quad y_i = y_i(\mathbf{X}; \underline{\mathbf{W}}) \tag{3}$$

zależnego od wag neuronów $\underline{\mathbf{W}}$ poprzez związki (1) i (2). Jedynymi adaptacyjnymi parametrami są zestawy wag, dla każdego neuronu zmieniające się w k-tym kroku uczenia według schematu:

$$\underline{\mathbf{W}}(k+1) = \underline{\mathbf{W}}(k) + \Delta\underline{\mathbf{W}}$$

tak, by maksymalnie upodobnić wyjście sieci do wartości oczekiwanej (czyli zminimalizować błąd (3) poprzez minimalizację różnicy pomiędzy odpowiedzią sieci na dany wzorzec a odpowiedzią prawdziwą). Szczegóły wielu metod uczenia zawarte są w [33].

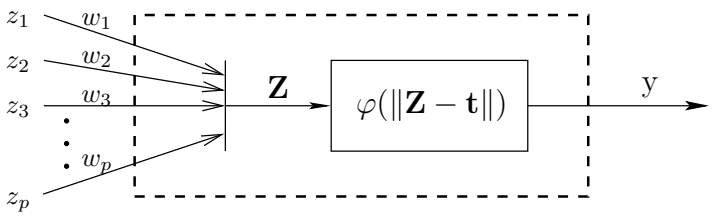


Rysunek 14: Granice decyzji dla sieci *MLP*.

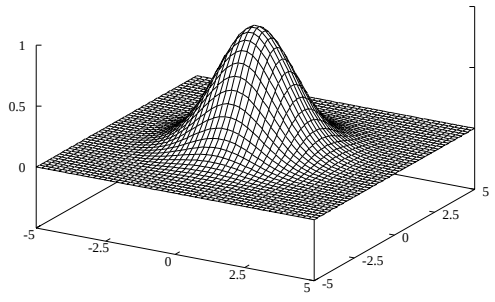
2.1.2 Sieć RBF

To, co w sieciach *MLP* powstawało przez wiele lat (zapewnienie nieliniowości tj. zdolności do nieliniowych odwzorowań), w sieciach *RBF* (***R**adial **B**asis **F**unction*) jest od razu wpisane w ich budowę. Wprowadzenie warstwy ukrytej z nieliniowymi neuronami realizującymi funkcje radialne wyeliminowało konieczność używania kilku warstw.

Sieci *RBF* stanowią odmienne, a zarazem uzupełniające, podejście do problemu interpolacji danych w stosunku do sieci *MLP*. Sieci *MLP* dokonują podziału przestrzeni za pomocą szeregu hiperpłaszczyzn, intuicyjnym wydaje się więc spróbować dokonać takiego samego podziału za pomocą hipersfer charakteryzowanych poprzez położenia centrów i promienie. Niezbędne jest zatem zdefiniowanie funkcji radialnych mierzących dystans danego punktu od ustalonego centrum. Na tym właśnie opiera się model neuronu dla sieci *RBF* (rysunek 15).



Rysunek 15: Schemat neuronu z warstwy ukrytej występującego w sieci *RBF*.



Rysunek 16: Gaussowska funkcja bazowa (4) dwóch zmiennych występująca w sieci *RBF* ($\sigma = 1.25$).

Najczęściej używaną funkcją radialną jest funkcja Gaussa:

$$\varphi(\|\mathbf{r}\|) = \exp\left(-\frac{\|\mathbf{r}\|^2}{2\sigma^2}\right) \tag{4}$$

przedstawiona na rysunku 16.

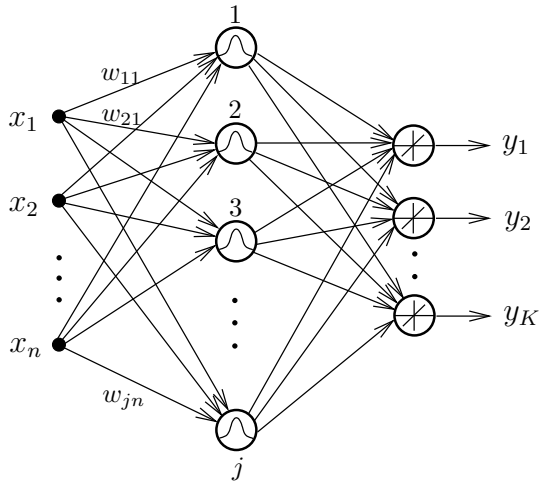
Dobrze rozwinięta jest teoria matematyczna opisująca interpolację funkcjami radialnymi. Z [33] wynika, że sieci *RBF* dokonują dokładnej aproksymacji funkcji, gdy liczba neuronów w warstwie ukrytej równa jest liczbie wzorców uczących. Ponieważ warunek ten praktycznie nigdy nie może być spełniony, potrzebna jest wstępna klasteryzacja danych, dzieląca wektory na grupy o wzajemnym podobieństwie. Liczba klastrów (symbolizowana przez liczbę neuronów warstwy ukrytej) powinna być równa lub większa od liczby klas zawartych w danych. Typową strukturę sieci *RBF* przedstawia rysunek 17.

Uczenie sieci składa się z dwóch etapów:

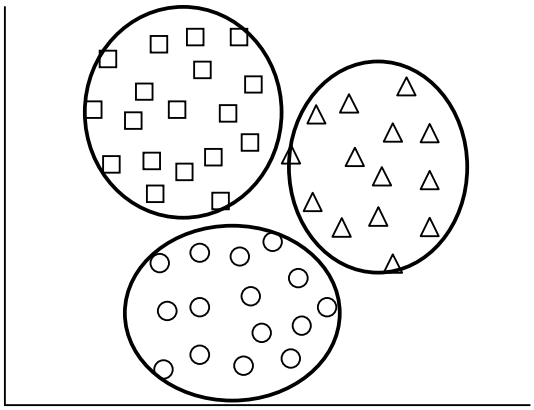
1. W pierwszym etapie dobierane są centra $\mathbf{t}$ i parametry kształtów funkcji bazowych. Jest to część trudniejsza, uczenie może przebiegać pod nadzorem lub być typu samoorganizującego.

2. Drugi etap to dobór wag neuronów warstwy wyjściowej – parametry te odpowiadają bezpośrednio za klasę do której zostanie przypisany nieznany wektor.

Granice decyzji sieci *RBF* są odmienne niż w przypadku *MLP*, istotne porównanie prezentują rysunki 14 i 18.



Rysunek 17: Ogólna postać sieci *RBF*. Krzywe Gaussa na neuronach symbolizują radialne funkcje bazowe.



Rysunek 18: Granice decyzji dla sieci *RBF*.

2.1.3 Sieć FSM

System *FSM* (***F**eature **S**pace **M**apping*), opisany w pracy [1], to narzędzie obliczeniowe oparte na architekturze sieci *RBF*. Patrząc z różnych punktów widzenia *FSM* może być nie tylko klasyfikatorem, lecz także systemem neurorozmytym, autoasocjacyjnym, jądrem systemu eksperckiego i wieloma innymi. Tu omówione zostaną jednak cechy systemu związane z klasyfikacją.

Sieć ma strukturę typowej sieci *RBF*: warstwa wejściowa, ukryta i wyjściowa (może być z jednym neuronem). Główną modyfikacją w stosunku do sieci

RBF są neurony ukryte – w sieci *FSM* realizują one dowolną funkcję separowalną, tj. spełniającą warunek[8]:

$$\varphi(\mathbf{X}) = \prod_k \varphi_k(x_k).$$

Ponadto każdy z neuronów posiada dodatkowe, wbudowane parametry. Takie założenia umożliwiają wprowadzenie nietypowych funkcji realizowanych przez neurony, m.in. trójkątnych, prostokątnych, trapezoidalnych – pozwala to na uzyskanie z sieci reguł logicznych.

Algorytm uczenia sieci jest algorytmem konstruktywistycznym – w czasie uczenia sieci zmienia się liczba neuronów w warstwie ukrytej. Sieć budowana jest od podstaw – na początku nie posiada ukrytych neuronów, choć ich liczba może być ustalona empirycznie lub też przez wbudowany algorytm klasteryzacji. W trakcie uczenia sieć, zgodnie z pojawiającymi się wzorcami uczącymi, analizuje konieczność dostawienia nowego neuronu – być może wystarczy tylko dokonać adaptacji parametrów. Po zakończeniu uczenia sieć przypisuje nowy przypadek do klasy, którą reprezentuje neuron o maksymalnej aktywacji:

$$FSM(\mathbf{X}^*) = \text{Klasa}\left[\max_k \varphi_k(\mathbf{X}^*)\right],$$

gdzie k jest indeksem numerującym neurony warstwy ukrytej.

Po każdorazowym przeanalizowaniu zbioru treningowego badana jest jakość klasyfikacji. Jeśli jest większa od zdefiniowanego progu, sprawdza się, czy uproszczenie struktury sieci (poprzez zmniejszenie liczby neuronów warstwy ukrytej) nie pogarsza wyników. Jeśli klasyfikacja nie jest wystarczająca, uczenie jest kontynuowane.

2.1.4 Sieć IncNet

Sieć *IncNet* (***Inc****remental* ***Net****work*) należy do grupy tzw. sieci ontogenicznych – mogących zmieniać swoją architekturę w procesie uczenia (dodając lub usuwając neurony według ustalonych zasad). Proces ten opiera się jednak na innych założeniach niż w sieci *FSM*. Tu złożoność sieci kontrolowana jest cały czas (w *FSM* tylko po prezentacji wszystkich wzorców) i mechanizmy autokontroli na bieżąco decydują o konieczności zmiany jej struktury (dodaniu, usunięciu lub połączeniu ze sobą neuronów). *IncNet* ma szczególne zastosowanie w klasyfikacji – działa wówczas w trybie tzw. K-klasyfikatora.

[8]Warunek ten spełnia także funkcja gaussowska postaci (4).

Zamiast rozseparować K klas za pomocą jednej sieci, problem zostaje podzielony na K podproblemów i do każdego z nich zostaje użyta oddzielna sieć, działająca na tych samych zasadach.

Budowa sieci *IncNet* opiera się na sieci *RBF*, istnieje jedna warstwa ukryta, z neuronami o różnych bicentralnych (kombinacja funkcji sigmoidalnych) lub gaussowskich funkcjach transferu. Tryb K-klasyfikatora ogranicza liczbę neuronów wyjściowych do jednego.

W uczeniu sieci wykorzystywany jest algorytm *EKF* (*Extended Kalman Filter*) – rozszerzony filtr Kalmana. Adaptacja wewnętrznych parametrów wiąże się z minimalizacją specjalnie określonej funkcji błędu [24]. Za zmianę struktury sieci odpowiedzialne są kryteria wyrażone przez zdefiniowane warunki i, w razie ich niespełnienia, realizujące je algorytmy (całość ma skutecznie „odpowiedzieć" na pytanie: „kiedy aktualna architektura sieci nie jest już wystarczalna, aby akumulować nowe informacje?"), których ścisłe przytoczenie tu, bez odpowiednio długiego wstępu, byłoby bezcelowe.

Po pojawieniu się nieznanego wektora, zostaje on przypisany do tej klasy, którą reprezentuje sieć z największą aktywacją neuronu wyjściowego.

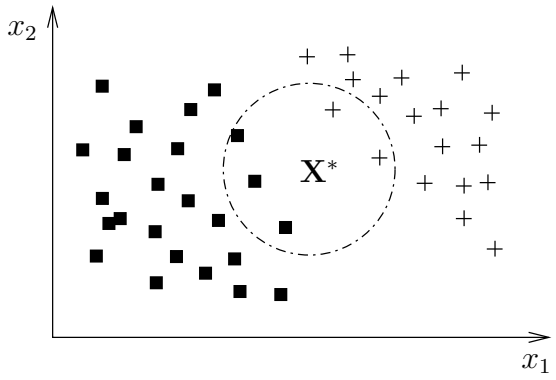
2.2 Klasyfikator kNN

Klasyfikator kNN (***k** **N**earest **N**eighbors*) jest przykładem klasyfikatora, który nie posiada żadnych parametrów adaptacyjnych – nie występuje w nim uczenie[9]. System kNN jest uogólnieniem metody *NN* ($k=1$) – najbliższego sąsiada – polegającej na przypisaniu nieznanego przypadku do najbliższego mu przypadku treningowego. Jest to podejście bazujące na zapamiętaniu wszystkich wzorców uczących – stąd bywa nazywane uczeniem poprzez zapamiętywanie lub też, biorąc pod uwagę sprawdzanie podobieństwa pomiędzy wektorami, uczeniem opartym na podobieństwie[10]. Wprowadzenie parametru k było konieczne ze względu na szum danych, który w wielu przypadkach wypacza wyniki – uwzględnienie nie jednego, lecz wielu przypadków często znacząco poprawia klasyfikację.

To z pozoru proste podejście ma liczne zalety: nie trzeba ustalać liczby klas, występuje duża stabilność metody (brak minimalizacji jakiejkolwiek funkcji nie prowadzi do lokalnych, suboptymalnych rozwiązań), brak adaptacyjnych

[9]Mimo to mówi się symbolicznie o uczeniu systemu kNN, by podkreślić w nim przetwarzanie dwóch zbiorów: treningowego i testowego.

[10]Jeszcze inna nazwa to, także uzasadnione, uczenie bezparametrowe [23].



Rysunek 19: Klasyfikacja modelu kNN dla $k = 5$. Nieznany wektor $\mathbf{X}^*$ zostaje przypisany do klasy „■", ponieważ większość z jego 5-ciu najbliższych sąsiadów reprezentuje tę klasę.

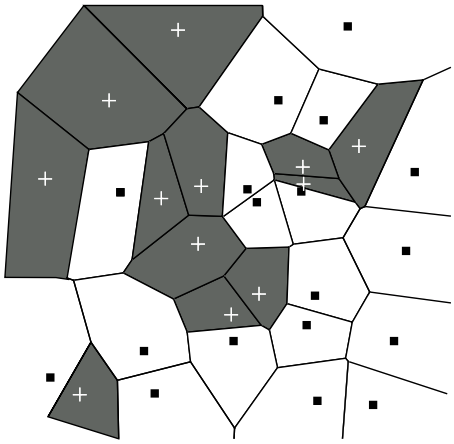
parametrów ogranicza przestrzeń poszukiwań. Do wad należą przede wszystkim: konieczność przechowywania dużej liczby danych w pamięci, duży koszt obliczeniowy dla większych zbiorów (także przy dużym n) i możliwe impasy dla parzystych k.

Jak wspomniano, proces klasyfikacji sprowadza się do odnalezienia k najbliższych sąsiadów dla wektora $\mathbf{X}^*$ i przypisania wektora do klasy reprezentowanej przez większość spośród wyszukanych wektorów (rysunek 19).

Wyznaczenie sąsiadów wymaga zatem określenia metryki w przestrzeni, na podstawie której obliczana będzie odległość. Do najczęściej stosowanych metryk należą[11]:

- Manhattan (d_1):
$$d(\mathbf{a}, \mathbf{b}) = \sum_{f=1}^{n} |a_f - b_f|.$$
- Euklidesowa (d_2):
$$d(\mathbf{a}, \mathbf{b}) = \left(\sum_{f=1}^{n} (a_f - b_f)^2 \right)^{1/2}.$$
- Chebysheva (d_∞):
$$d(\mathbf{a}, \mathbf{b}) = \max_{f=1}^{n} |a_f - b_f|.$$

[11]Pierwsze cztery metryki stanowią wspólną rodzinę funkcji Minkovskiego.



Rysunek 20: Dwuwymiarowe granice decyzji oraz obszary decyzyjne dla systemu kNN. Obserwujemy podział płaszczyzny na wieloboki Voronoia – symbolizujące wpływy poszczególnych wektorów treningowych.

- Minkovskiego (d_p):

$$d(\mathbf{a}, \mathbf{b}, p) = \left(\sum_{f=1}^{n} (a_f - b_f)^p \right)^{1/p}.$$

- Canberra:

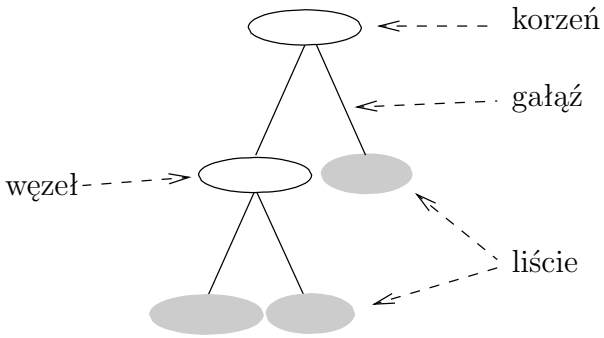
$$d(\mathbf{a}, \mathbf{b}) = \sum_{\substack{f=1 \\ a_f + b_f \neq 0}}^{n} \frac{|a_f - b_f|}{|a_f + b_f|}.$$

Mając ustaloną metrykę oraz parametr k można przystąpić do klasyfikacji. Granice decyzji dla modelu kNN mają charakter lokalny, ponieważ z idei metody wynika, iż dla konkretnego nieznanego przypadku nie jest szukana globalna relacja pomiędzy danymi, lecz tylko analizowane jego najbliższe otoczenie (otoczenie $\mathbf{X}^*$) – rysunek 20.

Udoskonalenia metody kNN polegają na automatycznym wyborze parametru k oraz metryki. Istnieją także odmiany kNN dokonujące selekcji i ważenia atrybutów.

2.3 Drzewa decyzji

Drzewa decyzji (*Decision Trees*) są przykładem klasyfikatora bazującym na tzw. metodach niemetrycznych – w procesie uczenia nie są w żadnym stopniu



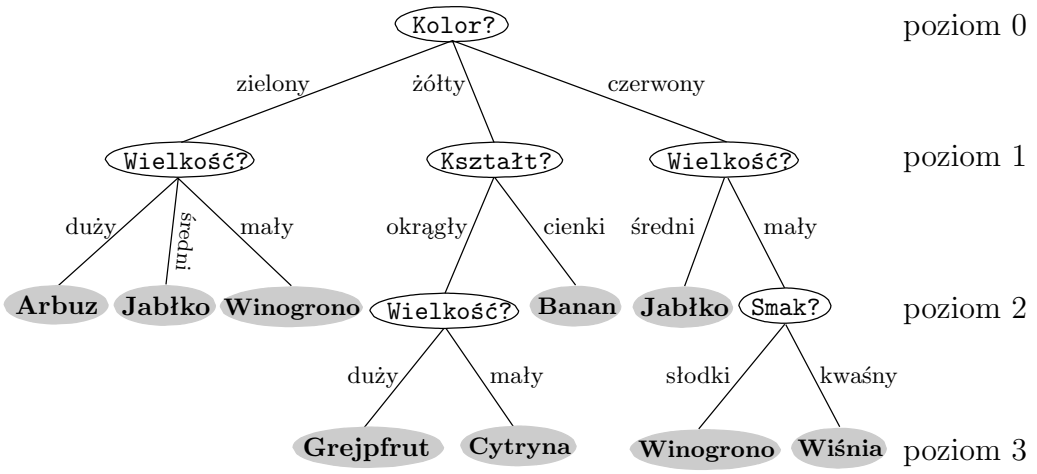
Rysunek 21: Struktura drzewa decyzji.

wykorzystywane informacje o wzajemnym podobieństwie obiektów (wektorów). Drzewa bazują na separacji cech – dzielą przestrzeń cech na części tak, by separacja obiektów z różnych klas była jak najlepsza. Teoretycznie więc drzewa mogą pracować na danych nienumerycznych – co ma wiele korzyści, lecz także wady[12]. W klasycznych drzewach decyzji odchodzi się od numerycznej reprezentacji cech na rzecz f-elementowych zbiorów atrybutów. Są to tzw. dane nominalne – skończony zbiór nieuporządkowanych wartości dyskretnych.

Struktura drzewa. Pomysł zapisu w postaci drzewa znany był już wcześniej w innych naukach (matematyka, biologia czy też informatyka). Terminologia elementów składowych DT jest zatem konsekwentnym stosowaniem nazewnictwa już ustalonego – nawiązujące do elementów tradycyjnego drzewa (rysunek 21). W skład typowego drzewa wchodzą więc następujące elementy:

- Korzeń (*root node*) – węzeł główny, do którego nie dochodzi żadna gałąź, rozpoczyna selekcję cech.
- Węzły (*nodes*) – odpowiadają testom na wartościach atrybutów.
- Gałęzie (*branches*) – odpowiadają możliwym wartościom wyników testów.
- Liście (*leaves*) – stanowią etykiety klas.

[12]Problem pojawia się, gdy napotykamy na dane ciągłe.



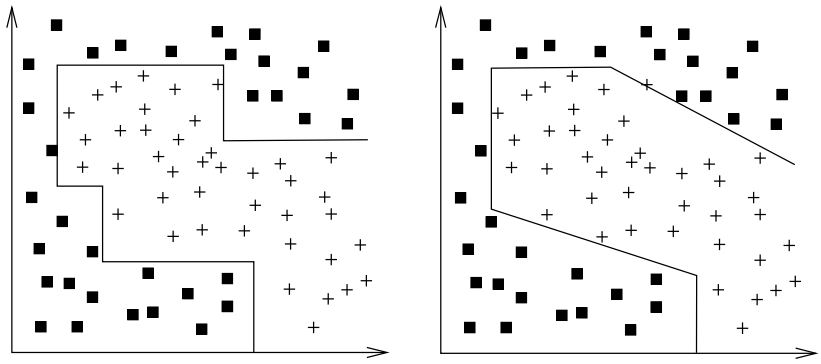
Rysunek 22: Działanie drzewa na prostym przykładzie (zbiór owoców).

Działanie drzewa. Węzeł główny (poziom 0) rozpoczyna budowanie drzewa, analizuje wartości wybranej cechy i tworzy odpowiednią liczbę gałęzi w poziomie pierwszym. Każda utworzona gałąź kończy się węzłem, na którym dokonują się podziały kolejnych cech. Węzły te są więc *de facto* węzłami głównymi dla nowopowstałych poddrzew. Podział przebiega aż do osiągnięcia poziomu liści – nie ma wówczas kryteriów dalszych podziałów. Klasyfikacja nieznanego przypadku polega na analizie atrybutów badanego wektora (poprzez serię pytań zawartych w węzłach), aż do momentu jego dojścia do jednego z liści. Wówczas obiektowi zostaje przypisana klasa, którą dany liść reprezentuje.

Dużą przewagę drzew decyzji nad innymi klasyfikatorami stanowi łatwość interpretacji ich wyników. Rezultaty klasyfikacji zapisać można w postaci logicznych reguł, które w prosty sposób pozwalają klasyfikować nowe przypadki oraz definiują klasy poprzez określenie warunków, jakie muszą spełniać obiekty do nich należące.

Przykład
Rozważmy proste drzewo do problemu klasyfikacji owoców, każdy obiekt opisywany będzie 4 cechami: `smak`, `kolor`, `kształt` i `wielkość`. Utworzone drzewo przedstawia rysunek 22. Nieznany przypadek $\mathbf{X}^* = \{$`słodki`, `żółty`, `cienki`, `średni`$\}$ zostanie sklasyfikowany jako obiekt klasy `banan`, ponieważ spełnia regułę[13]: `(kolor = żółty) AND (kształt = cienki)`. Jak widać,

[13]Użyte oznaczenia: `AND`, `OR` i `NOT` to dobrze znane operatory logiczne oznaczające odpowiednio koniunkcję, alternatywę oraz zaprzeczenie zdania.



Rysunek 23: Granice decyzji dla drzewa. Po lewej stronie klasyczne drzewo z testem tylko jednego atrybutu na węźle. Po prawej drzewo z testem obydwu atrybutów na wybranych węzłach – obserwujemy skośne granice decyzji.

prostą analizę pojedynczego wektora można przeprowadzić „ręcznie", co nie jest możliwe w przypadku choćby sieci neuronowych. Także utworzenie definicji klas nie jest zadaniem trudnym, przykładowo obiekt należący do klasy jabłko spełniać musi reguły: `(zielony AND średni) OR (czerwony AND średni)` lub: `średni AND NOT żółty`.

Proste drzewa produkują proste granice decyzji (rysunek 23), często jednak potrzebne są struktury zdolne do bardziej skomplikowanych odwzorowań. Dlatego też powstało wiele odmian drzew różniących się budową, testowaniem atrybutów lub sposobem ustalania liści. Jednym z bardziej powszechnych (i zarazem prostszych) jest drzewo binarne, w którym z każdego węzła wychodzą dwie gałęzie (zamiast liczby gałęzi równej liczbie różnych atrybutów – jak na rysunku 22). Reprezentują one przeciwstawne odpowiedzi na test wartości atrybutu badanego na węźle, wynikiem mogą być tylko dwie odpowiedzi „tak" lub „nie", np. `kolor=żółty?`. Innym pomysłem jest drzewo, w którego węzłach bada się nie jeden, lecz kilka atrybutów, wówczas w zastosowaniu do danych numerycznych granice decyzji mają inny charakter (rysunek 23).

Dla objętościowo dużych danych budowane drzewa mają złożoną strukturę, reguły natomiast stają się skomplikowane. Dodatkowo pojawia się szereg nowych problemów, których rozwiązanie (lub próba rozwiązania) ma znaczący wpływ na wynik klasyfikacji. Oto najważniejsze z nich (ilustrujące zarazem złożoność tematyki DT i sugerujące duże bogactwo rozwiązań):

1. Generalizacja i przycinanie (*prunning*) drzewa.

   Każde drzewo może idealnie dopasować się do danych, tworząc wyolbrzymioną strukturę uwzględniającą poszczególne (pojedyncze) przypadki treningowe – mamy wówczas typowe dopasowanie się modelu do szumu danych (w skrajnym wypadku każdemu przypadkowi treningowemu może odpowiadać jeden liść!). Taka struktura nie może dobrze generalizować. Konieczne jest podjęcie decyzji o przycinaniu gałęzi drzewa, uproszczeniu jego struktury, by, mimo braku czystości węzłów (przez dany węzeł będą przechodziły także błędne przypadki), końcowa generalizacja była optymalna. Tu pojawiają się następujące pytania:

   - Jakie algorytmy przycinania stosować?
   - Kiedy węzeł powinien stać się liściem?
   - Jaka czystość węzła jest dopuszczalna?

2. Problem danych ciągłych.

   Klasyczne drzewo dokonuje podziału wartości dyskretnych, zbiór możliwych odpowiedzi jest zatem ograniczony. Próba zastosowania drzewa do danych ciągłych wymaga ustalenia kryterium podziału przedziału (a, b) na określoną liczbę podprzedziałów. Najprostszym rozwiązaniem wydaje się podział na dwie części $(a, \frac{b-a}{2})$ i $(\frac{b-a}{2}, b)$. Istnieje jednak wiele innych sposobów, np. metody starające wybrać się taki punkt podziału, by maksymalnie odseparować wektory z różnych klas.

3. Która cecha powinna zostać przetestowana na węźle głównym?

   Często wybór innej cechy jako analizowanej w pierwszej kolejności może całkowicie zmienić reguły, także bardzo je uprościć.

Istnieje wiele algorytmów konstrukcji drzew próbujących uporać się z wyżej wymienionymi problemami (*CART*, *ID3*, *C4.5* opisanych w [14]), większość z nich działa zgodnie z tzw. brzytwą Okhama, tzn. preferuje proste rozwiązania dające dobre wyniki (zatem drzewo powinno mieć jak najmniej skomplikowaną budowę i produkować w miarę czytelne reguły).

2.3.1 Drzewo SSV

Drzewo decyzji *SSV* (***S**eparability of **S**plit **V**alue*) to klasyczne drzewo z testem jednego atrybutu na węźle, w którym zastosowano nowatorskie rozwią-

zanie podziału wartości cech na dwa przedziały. Drzewo to może być użyte zarówno do danych ciągłych, jak i dyskretnych. Algorytm *SSV* wyznacza optymalny podział danych tak, by separowalność wektorów z różnych klas była możliwie największa. Każdy przedział wartości (lub zbiór danych) cech zostaje podzielony na dwie części – dlatego z każdego węzła drzewa wychodziły będą dwie gałęzie.

Głównym pojęciem algorytmu jest wartość dzieląca s (*split value*), która dla danych ciągłych jest liczbą rzeczywistą, dla dyskretnych podzbiorem (może być jednoelementowym). W obu przypadkach, ustalona wartość s dzieli zbiór wektorów na dwa podzbiory, umownie nazwane lewą (LS) i prawą (RS) stroną danego s (dla cechy f i zbioru danych $\mathcal{X}$):

$$\begin{aligned} \mathrm{LS}(s, f, \mathcal{X}) &= \begin{cases} \{\mathbf{X} \in \mathcal{X} : f(\mathbf{X}) < s \quad \} & \leftarrow \text{cechy ciągłe} \\ \{\mathbf{X} \in \mathcal{X} : f(\mathbf{X}) \notin s \quad \} & \leftarrow \text{cechy dyskretne} \end{cases} \\ \mathrm{RS}(s, f, \mathcal{X}) &= \mathcal{X} - \mathrm{LS}(s, f, \mathcal{X}), \end{aligned}$$

gdzie $f(\mathbf{X})$ jest wartością f-tej cechy wektora $\mathbf{X}$. Mając wartość s można stwierdzić do jakiej separowalności ona prowadzi:

$$\mathrm{SSV}(s) = 2 * \sum_{c \in \mathcal{C}} |LS(s, f, \mathcal{X}) \cap \mathcal{X}_c| * |RS(s, f, \mathcal{X}) \cap (\mathcal{X} - \mathcal{X}_c)|$$

$$- \sum_{c \in \mathcal{C}} \min(|LS(s, f, \mathcal{X}) \cap \mathcal{X}_c|, |RS(s, f, \mathcal{X}) \cap \mathcal{X}_c|),$$

$\mathcal{C}$ jest zbiorem wszystkich klas, a $\mathcal{X}_c$ zbiorem wektorów należących do klasy c ($c \in \mathcal{C}$).

Budowa drzewa rozpoczyna się od węzła głównego. Obliczamy separowalność dla każdego z możliwych s, wybieramy s dające najlepszy podział i powtarzamy procedurę dla nowopowstałych węzłów, dla każdego szukając optymalnego podziału. Wybór optymalnej wielkości dzielącej może być dokonywany dwoma sposobami:

- Przeszukiwanie „najpierw najlepszy” (*best first search*) – ograniczamy się do wyboru takiego s, które prowadzi do maksymalnej separacji na danym węźle. Nie korzystamy jednak z informacji jakie będą separowalności na następnych węzłach.
- Przeszukiwanie wiązką (*beam search*) – nie ograniczamy się tylko do nowo utworzonego poziomu, przeprowadzamy także analizę kolejnych podziałów (poziomów). Strategia ta jest bardziej długodystansowa, powinna więc dawać lepsze wyniki – wszystko to kosztem czasu i rosnącej złożoności obliczeniowej.

Dodatkowo celem osiągnięcia możliwie najlepszej generalizacji w drzewie stosowane są algorytmy przycinające.

Więcej informacji na temat *SSV* oraz otrzymanych wyników znajduje się w [10, 21].

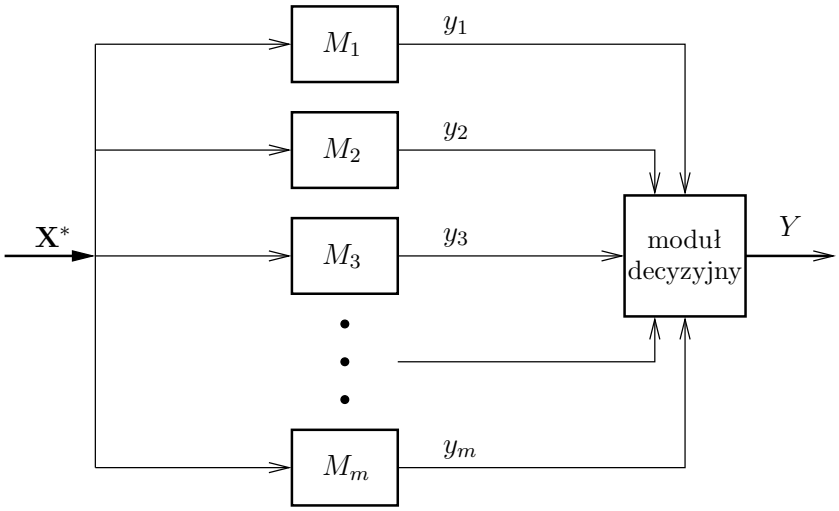
3 Komitety

3.1 Dlaczego komitet?

Co dwie głowy to nie jedna, głosi stare przysłowie, i zapewne pierwszym twórcom komitetów nie było ono obce. Kiedy rzeczywistość, wraz ze stawianymi problemami, zaczyna przerastać istniejące rozwiązania i wyprzedzać szybkość maszyn, pojawia się potrzeba nowych rozwiązań, mających, w perspektywie, zapewnić lepsze wyniki. W takich chwilach jedni podążają w kierunku całkowicie nowych rozwiązań (np. tworząc nowe klasyfikatory), nie patrząc w przeszłość, inni koncentrują się na teraźniejszości i chcą wydobyć z istniejących pomysłów jak najlepsze wyniki. Jednym z szybszych i mniej kosztownych sposobów na zwiększenie wydajności istniejących już rozwiązań stały się komitety systemów klasyfikujących.

Komitet (*Ensemble, Committee, Multiple Classifier Systems – MCS*) to symboliczna nazwa zbioru klasyfikatorów (modeli), które, pracując wspólnie nad tym samym zadaniem, mają dać wynik lepszy, niż wyniki otrzymywane przez pojedyncze modele. Ten początkowo prosty pomysł stał się popularny na początku lat 90-tych, powstał podczas prac nad rozpoznawaniem pisma ręcznego i dotyczył głównie sieci neuronowych. Dziedzina zaczęła się szybko rozwijać (stymulowana obiecującymi wynikami) i, pokonując okres tworzenia komitetów złożonych z modeli tego samego rodzaju (np. wyłącznie sieci neuronowe lub drzewa decyzji), przeobraziła się pole badawcze pełne niezwykle wyrafinowanych pomysłów.

Pojedyncze modele często są niestabilne (małe zaburzenie zbioru treningowego może prowadzić do całkowicie innych granic decyzji), stochastyczne elementy w metodach uczenia (np. przypadkowa inicjalizacja wag początkowych sieci neuronowej) w połączeniu z procedurą minimalizacyjną funkcji błędu, często dają różne wyniki. Komitet ma być pozbawiony tych słabości, jest mniej podatny na zaburzenia i to stanowi jego przewagę. Ponadto, tak jak w codziennym życiu, także w uczeniu maszynowym nie ma narzędzi uniwersalnych. W projekcie Statlog [30], oceniającym różne rodzaje klasyfikatorów poprzez analizę wielu zbiorów danych, wykazano, iż nie ma uniwersalnych systemów klasyfikujących, działających tak samo dobrze (lub w stopniu zbliżonym) na każdym zbiorze danych. Wręcz przeciwnie, dla każdego z modeli znaleźć można zbiory dające lepsze i gorsze wyniki, a w ramach tego samego zbioru danych obszary lepszej lub gorszej klasyfikacji. Tu uwidacznia się kolejny atut komitetu, w porównaniu z pojedynczymi klasyfikatorami; potrafi on – przede wszystkim dzięki bardziej rozbudowanej strukturze – lepiej



Rysunek 24: Ogólny schemat budowy i działania komitetu składającego się z m klasyfikatorów. Nieznany wektor $\mathbf{X}^*$ zostaje przeanalizowany przez wszystkich członków komitetu podejmujących własne decyzje y. Końcowy rezultat klasyfikacji Y jest zawsze wynikiem założonej filozofii działania komitetu realizowanej przez moduł decyzyjny.

dopasować się do złożonej struktury analizowanych danych.

3.2 Podstawowe informacje

Budowa komitetu

Typowa struktura komitetu przedstawiona jest na rysunku 24. Zbiór m klasyfikatorów podejmuje własne decyzje, główna część komitetu to część decyzyjna, tworząca głos końcowy na podstawie głosów członków komitetu (często proces ten nazywamy kombinowaniem rezultatów).

Niezależnie od rodzaju komitetu zawsze następuje przetworzenie danych wejściowych $\mathcal{X}$ i podjęcie pojedynczych decyzji y. Procesy prowadzące od danych $\mathcal{X}$ do głosu Y w różnych rodzajach komitetów są całkowicie odmienne.

Rodzaje komitetów

Bogactwo proponowanych praktycznych rozwiązań w dziedzinie przełożyło się na teoretyczne (i często tylko symboliczne – wielu przypadków nie da się

jednoznacznie określić) zróżnicowanie komitetów. Ze względu na rozmaite kryteria możemy je więc podzielić na:

1. Statyczne lub dynamiczne (*combination* lub *selection*).
   W komitetach statycznych wszystkie modele mają jednakowy wkład do końcowej decyzji, w dynamicznych natomiast wybieramy zawsze jeden najbardziej kompetentny model, często nazywany ekspertem, który podejmuje decyzję za wszystkich.

2. Komitety oparte na głosowaniu (*voting*) lub prawdopodobieństwach (*non-voting*).
   W głosowaniu zliczamy końcowe decyzje każdego modelu lub tworzymy ich kombinację liniową (może być ważona) i na jej podstawie tworzymy końcowy głos. W metodach *non-voting* zamiast na decyzjach bazujemy na prawdopodobieństwach $P(C|\mathbf{X})$ i na ich podstawie podejmujemy decyzję.

3. Komitety modeli niezależnych lub uczonych równocześnie.
   Każde z podejść ma swoje zalety i wady:

   - W pierwszym przypadku każdy model wchodzący w skład komitetu jest tworzony i trenowany niezależnie – bez świadomości istnienia pozostałych modeli, jego głos (końcowa klasyfikacja) zostaje włączony do ostatecznej decyzji. Umożliwia to budowanie komitetów z klasyfikatorów różnych typów, nawet takich, w których nie występuje uczenie. Minusem takiego podejścia jest niebezpieczeństwo, iż każdy model może wyspecjalizować się w tym samym obszarze danych – nie mając informacji, iż wcześniej dokonał tego inny model.
   - W uczeniu równoczesnym modele wzajemnie oddziaływują na siebie podczas procesu uczenia (zamiast uczenia pojedynczych modeli mamy *de facto* uczenie całego komitetu połączone z uczeniem modeli). Teoretycznie to podejście powinno zapewniać lepsze dopasowanie się do danych i zapobiec braku równomierności w tym procesie. Zmuszeni zostajemy jednak do używania modeli jednego rodzaju (np. tylko sieci neuronowe), z reguły proces uczenia takiej struktury jest także bardzo długotrwały.

4. Komitety homo- lub heterogeniczne – dzielimy komitety na składające się z klasyfikatorów tego samego rodzaju lub nie wprowadzamy żadnych ograniczeń. Pierwsze z podejść, wywodzące się z początków dziedziny, jest do dziś zdecydowanie powszechniejsze.

5. Ze względu na sposób przetwarzania danych rozróżniamy komitety, w których każdy z członków w fazie uczenia pracuje na tym samym zbiorze danych, oraz takie, w których pojedyncze klasyfikatory pracują na różnych podzbiorach jednego zbioru głównego.

Tworzenie komitetu

Komitet składa się z pojedynczych modeli, które powinny działać tak, by całościowa klasyfikacja komitetu była lepsza niż decyzje indywidualne członków. Nie oznacza to, że dobór członków komitetu może być przypadkowy. Innym problemem jest to, ilu spośród członków i w jaki sposób powinno podejmować ostateczną decyzję. Tworzenie komitetu, można zatem podzielić na dwa etapy: tworzenie struktury komitetu i tworzenie funkcji decyzyjnej. Trudno stwierdzić, który z etapów jest ważniejszy, większość współczesnych opracowań skupia się jednak na etapie drugim, czyli na stworzeniu optymalnego modułu decyzyjnego.

Dobierając odpowiednich członków komitetu, pierwszą rzeczą, którą można stwierdzić jest to, iż tworzenie komitetu z identycznie działających modeli jest bezużyteczne. Dlatego szczególnie ważnym zagadnieniem jest to, jakie modele oraz jaka ich liczba powinna utworzyć komitet. Często należy kierować się intuicją (co ma miejsce w rzeczywistości), rozsądek nakazuje zwrócenie uwagi na trzy zagadnienia:

- Różnorodność – cecha bardzo potrzebna. Jeśli w komitecie występują klasyfikatory dokonujące identycznych klasyfikacji dla wybranych podzbiorów danych, nie jest możliwe, by wynik komitetu mógł być lepszy od wyniku pojedynczego modelu. Tylko tam, gdzie występują błędy modeli w różnych miejscach – symbolizowanych przez podprzestrzenie przestrzeni cech – można otrzymać zwiększoną dokładność klasyfikacji (choć całkowita dokładność poszczególnych modeli może być jednakowa). Istnieją różne sposoby szacowania różnorodności modeli: definicja mówi, iż dwa klasyfikatory są różne, jeśli popełniają różne błędy dla nowych danych. Bardzo użytecznym narzędziem badania różnorodności jest także macierz konfuzji.

- Wydajność – działanie komitetu musi być szybkie, nie pochłaniać dużej ilości czasu i mocy obliczeniowej. Samo nauczenie i działanie pojedynczych klasyfikatorów jest już zabiegiem kosztownym, dlatego komitet nie powinien znacząco zwiększać czasu obliczeń.

- Dokładność – wspomniane już wcześniej założenie, iż ogólny wynik komitetu powinien być lepszy niż wynik klasyfikacji poszczególnych członków.

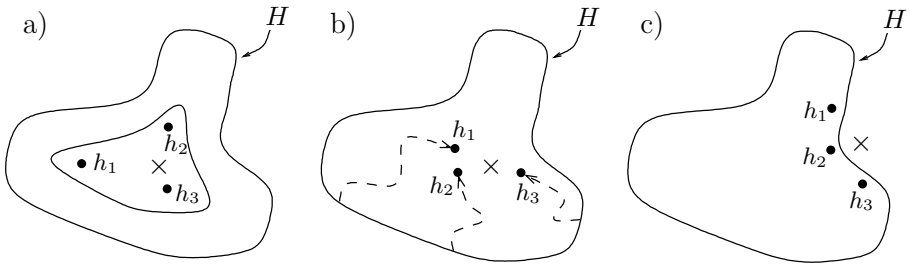
Wymienione cechy zazwyczaj trudno pogodzić, w praktyce jednak, największą uwagę należy zawsze zwracać najpierw na różnorodność, później na dokładność[14]. Dodatkowo należy obserwować zachowanie się modeli w poszczególnych regionach (co nie zawsze widać w macierzy konfuzji), stabilność modeli jest generalnie pożądana jednak komitety potrafią sobie dość dobrze radzić z jej obniżonym poziomem.

Kolejnym ważnym zagadnieniem jest dobór liczby członków komitetu. Duża liczba modeli to wzrastające koszty obliczeniowe, zarówno działania komitetu, jak i uczenia modeli. Zbyt mała ich liczba może nie zapewnić pożądanej poprawy dokładności. Generalnie powinno się preferować jak najmniejszą liczbę modeli, a rozwiązanie powinno być kompromisem pomiędzy czasem uczenia komitetu a jego dokładnością. W pracach nad rozpoznawaniem pisma ręcznego za pomocą sieci neuronowych okazało się, iż wystarczająco znaczącą poprawę klasyfikacji uzyskiwano dla dwóch lub trzech sieci wchodzących w skład komitetu. Podobne wyniki ujawniły prace z wykorzystaniem drzew decyzji do przewidywania prognozy pogody. Również w oryginalnych pracach nad algorytmem *boosting* wykorzystywano trzy modele. W wielu innych projektach znaczące wyniki otrzymywano, gdy liczba klasyfikatorów nie przekraczała 10-ciu. Niejako dla kontrastu prowadzi się badania nad komitetami zbudowanymi z olbrzymiej liczby klasyfikatorów (nawet do 100 sieci neuronowych lub 200 drzew decyzji) jednak wydają się one mieć znaczenie bardziej teoretyczne niż użyteczne. W codziennej praktyce liczbę członków komitetu wciąż najczęściej dobiera się metodą prób i błędów, dostosowując strukturę komitetu do konkretnego problemu.

Działanie komitetu

Pomimo oczywistego i nienegowanego stwierdzenia, iż grupa systemów powinna działać lepiej niż każdy z nich osobno, istnieją próby bardziej racjonalnego podejścia do tego zagadnienia, udowodnienia i wytłumaczenia działania komitetu. Zrozumienie takiego podejścia ułatwia wprowadzenie pojęcia „hipoteza". W klasyfikacji dany model podejmuje decyzję o klasie wektora $\mathbf{X}^*$, która, z racji braku stuprocentowej pewności, jest tylko hipotezą – $h(\mathbf{X}^*)$.

[14]W pracy [35] opisano dwa pojedyncze klasyfikatory o dokładności 23% i 25%, które, po stworzeniu komitetu, osiągnęły razem 69%. Dodatkowo *boosting* zwiększył ten wynik do 88%.



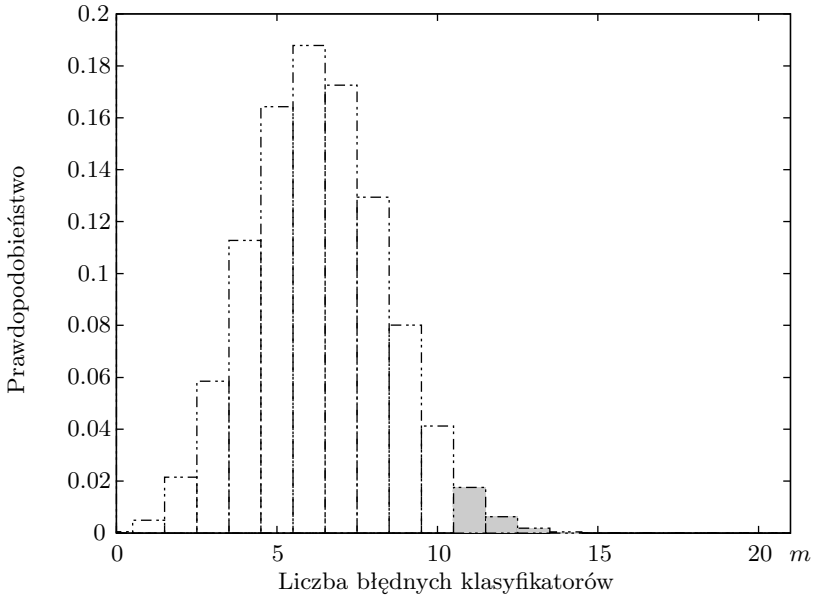
Rysunek 25: Trzy podstawowe powody działania komitetu: a) statystyczny, b) obliczeniowy, c) reprezentacyjny. h oznacza hipotezę w przestrzeni H, natomiast „×" jest prawdziwą hipotezą.

Inny klasyfikator może podjąć inną decyzję (lub taką samą, ale z mniejszą pewnością) – otrzymujemy nową hipotezę. Wprowadzić można pojęcie przestrzeni hipotez H oraz hipotezę „prawdziwą" („×"), czyli poszukiwaną zależność – będą one użyteczne w zaprezentowaniu trzech powodów dlaczego prosty komitet powinien mieć przewagę nad pojedynczym modelem. Według [9] istnieją trzy takie powody (rysunek 25):

- Statystyczny – proces uczenia może być interpretowany jako przeszukiwanie przestrzeni hipotez w celu znalezienia najlepszej z nich. W przypadku, gdy nie ma dostatecznie licznego zbioru treningowego lub też wymiary przestrzeni H są zbyt duże, pojedynczy algorytm znaleźć może wiele hipotez h dających te same wyniki na zbiorze treningowym. Uśredniając wyniki kilku klasyfikatorów zmniejszamy ryzyko wyboru błędnej hipotezy.

- Obliczeniowy – uczenie lub działanie klasyfikatora to ustalanie pewnych relacji w przestrzeni cech (lub też w przestrzeni hipotez). W przypadku obszernych zbiorów danych, zwykle zbadanie całej przestrzeni przez pojedynczy model nie jest niemożliwe – problemem jest nie tylko czas obliczeń, ale także duże ryzyko utknięcia procesu obliczeniowego w minimum lokalnym. Jednak zbiór klasyfikatorów badający różne obszary przestrzeni, może ją przeszukać dokładniej i znaleźć optymalne rozwiązanie szybciej.

- Reprezentacyjny – klasyfikatory opierają swe działanie na informacji wydobywanej ze zbioru treningowego – na jej podstawie stawiają hipotezy. Zdarzyć się może, iż prawdziwa hipoteza znajduje się poza dostępnym obszarem przestrzeni H. Komitety pozwalają złamać to ogra-



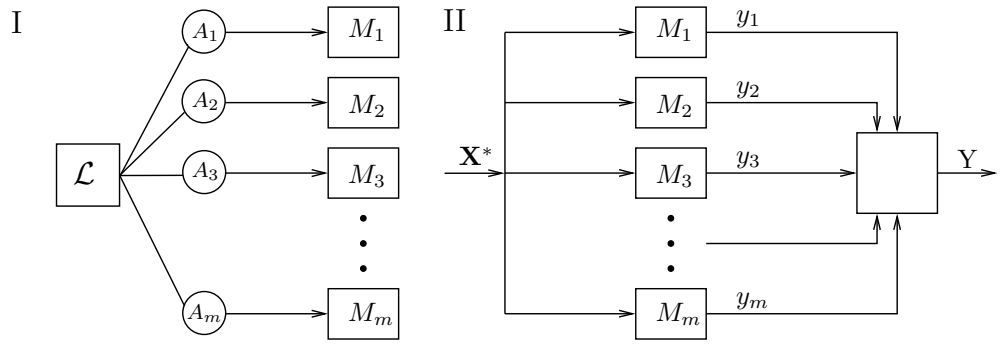
Rysunek 26: Prawdopodobieństwo popełnienia błędów przez dokładnie określoną liczbę (m) klasyfikatorów. W przypadku głosowania większościowego ($m \geqslant 11$) wynosi ono 0.026 i jest dużo mniejsze, niż prawdopodobieństwo popełnienia błędu przez pojedynczy model (wynoszące dla każdego modelu 0.3).

niczenie – ważona suma hipotez może dokonać pewnego rodzaju ekstrapolacji poza przestrzeń H.

Jeden z matematycznych dowodów dlaczego komitet może działać lepiej niż pojedyncze modele, opiera się na rozkładzie dwumianowym prawdopodobieństwa, zwanym także rozkładem Bernoulliego. Rozkład ten wyraża prawdopodobieństwo P wystąpienia k sukcesów w n niezależnych próbach, w których każdy sukces może wystąpić z prawdopodobieństwem p:

$$P(n,k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{5}$$

Jeśli mamy m klasyfikatorów i błąd każdego z nich jest mniejszy niż $\frac{1}{2}$ oraz błędy są niezależne, wówczas prawdopodobieństwo, że większość klasyfikatorów podejmie błędną decyzję, będzie opisywane polem pod krzywą rozkładu dwumianowego, gdzie więcej niż $\frac{m}{2}$ hipotez jest błędnych. Przykład dla 21 klasyfikatorów przedstawia rysunek 26.



Rysunek 27: Schemat komitetu wykorzystującego próbkowanie danych. Działanie komitetu składa się z dwóch etapów, I: generowanie klasyfikatorów (za pomocą algorytmów A) działających na podzbiorach zbioru treningowego $\mathcal{L}$, II: klasyfikacja, czyli podejmowanie końcowej decyzji o wektorze $\mathbf{X}^*$.

3.3 Rodzaje komitetów

Mnogość i zróżnicowanie rozwiązań nie sprzyja podziałom komitetów, gdyż często nie da się dokonać podziałów rozłącznych. Zaproponowano więc umowny podział na trzy grupy zawierające opis założeń danego zbioru oraz bardziej szczegółowe omówienie konkretnych rozwiązań. Podrozdział czwarty, krótko charakteryzuje pozostałe rozwiązania. Wszędzie przez m oznaczono liczbę członków komitetu.

3.3.1 Statystyczne próbkowanie danych

Metody bazujące na statystycznym próbkowaniu (nazywanego także manipulowaniem) zbioru treningowego polegają na generowaniu serii klasyfikatorów pracujących na różnych podzbiorach zbioru głównego (rysunek 27). Metody te polecane są zwłaszcza dla niestabilnych algorytmów (praca na różnych podzbiorach ma wykluczyć znaczący wpływ przypadkowych zmian w zbiorze głównym na wynik), często wspomina się, iż polecane są dla algorytmów działających słabo (*weak learner*) – dla nich też osiągają najlepsze efekty.

Generowanie modeli wymaga bezpośredniego dostępu do algorytmu uczącego, jeśli jest on kosztowny obliczeniowo, koszt działania komitetu będzie do niego proporcjonalny. Nie występuje jednak jakakolwiek ingerencja w algorytm uczący, zróżnicowanie opiera się na zbiorach uczących, na których działają poszczególne algorytmy. Dużym atutem takiego rozwiązania jest możliwość zastosowania wszystkich dostępnych algorytmów.

Po zakończeniu fazy uczenia komitetu, czyli wygenerowaniu określonej liczby klasyfikatorów, komitet jest gotowy do łącznej klasyfikacji. Polega ona na wygenerowaniu pojedynczych głosów i ustaleniu na ich podstawie jednego – końcowego. Ten etap nie będzie jednak analizowany w tym podrozdziale[15].

Bagging

Jest to jeden z najprostszych pomysłów [4], który opiera się na idei wprowadzonej w *bootstrappingu* (stąd nazwa ***B**ootstrap **Agg**regation*). Z głównego zbioru danych generujemy kilka podzbiorów i uczymy każdy ze składników komitetu na oddzielnym podzbiorze. Dla m klasyfikatorów oraz zbioru uczącego $\mathcal{L}$ składającego się z N wektorów tworzy się m zbiorów uczących, w każdym znajduje się N wektorów losowo wybranych z $\mathcal{L}$. Prawdopodobieństwo wyboru każdego wektora jest równe i wynosi $\frac{1}{N}$, w niektórych zbiorach pewne wektory mogą być umieszczone więcej niż raz, inne wcale. Każdy model uczony jest według tego samego algorytmu, każdy proces uczenia przebiega niezależnie. Końcowa decyzja dla zagadnienia klasyfikacji podejmowana jest przez głosowanie większościowe.

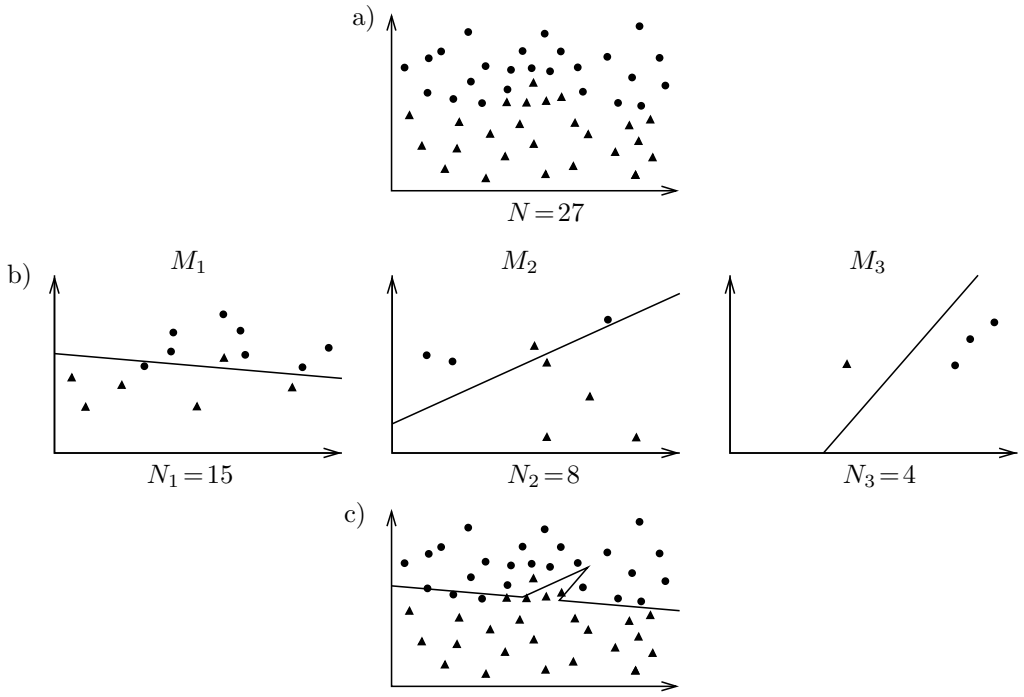
Boosting

Opisany w [16], jeden z najwydajniejszych i najczęściej stosowanych pomysłów mający na celu zwiększenie dokładności dowolnego algorytmu uczenia. Od *baggingu* różni się tym, iż następny klasyfikator pracuje na błędach pozostałego. Wektory źle klasyfikowane przez poprzedni model mają większe znaczenie w procesie uczenia następnego modelu, każdy proces uczenia przebiega niezależnie.

Początkowa wersja komitetu zakładała istnienie trzech modeli. Uczenie zaczynamy od modelu M_1 i filtrujemy zbiór danych treningowych tak, by nauczyć na utworzonych podzbiorach dwa dodatkowe modele: M_2 i M_3. M_1 jest modelem głównym, którego klasyfikację chcemy zwiększyć, kolejne kroki algorytmu są następujące:

1. Trenujemy M_1 na zbiorze treningowym $\mathcal{L}$. Uczenie przebiega tak długo, aż napotkany zostanie pierwszy wektor niedający się poprawnie sklasyfikować (wystąpi błąd).

[15] Kombinacje końcowych głosów zostają szczegółowo omówione w podrozdziale następnym.



Rysunek 28: Zastosowanie procedury *boosting* dla dwuklasowego zbioru danych (a). Dane zostają podzielone na zbiory treningowe dla poszczególnych modeli (b), w konsekwencji utworzona zostaje nieliniowa granica decyzyjna (c). N oznacza liczbę wektorów wchodzących w skład zbioru.

2. Wówczas błędny wektor umieszczamy w zbiorze treningowym drugiego modelu. Wracamy do punktu pierwszego i powtarzamy ten etap tak długo, aż w zbiorze treningowym drugiego modelu znajdzie się dostatecznie duża liczba wektorów.

3. Rozpoczynamy uczenie modelu M_2.

4. Klasyfikujemy pozostałe wektory przez modele M_1 i M_2, jeśli otrzymujemy sprzeczne wyniki dla danego wektora, umieszczamy go w zbiorze treningowym dla modelu M_3.

5. Zgromadzenie dostatecznie dużej liczby wektorów dla modelu M_3 rozpoczyna jego proces uczenia. Wraz z zakończeniem uczenia modelu M_3 uczenie komitetu zostaje zatrzymane.

Ostateczna decyzja zostaje podjęta przez głosowanie większościowe i prowadzi do kombinowanych granic decyzji (rysunek 28).

AdaBoost

Popularną odmianą *boostingu* jest *AdaBoost* (***Ada****ptive* ***Boost****ing*), algorytm, w którym przechowujemy serię wag przypisanych do każdego wektora ze zbioru treningowego. Wagi tworzą zbiór $\mathcal{D}_t$ – jest to główna cecha algorytmu. Początkowo waga każdego wektora jest równa, w miarę postępowania procesu uczenia wagi wektorów klasyfikowanych błędnie zostają zwiększane. Na podstawie zbioru wag tworzymy nowy zbiór danych (np. poprzez odrzucenie wektorów, których wagi nie przekraczają pewnego progu) – w ten sposób kolejny uczony klasyfikator może skoncentrować się na trudniejszych (z punktu widzenia klasyfikacji) przypadkach.

W podstawowej wersji algorytmu rozważany jest przypadek dwuklasowy (klasy oznaczamy jako $+1$ i -1). Algorytm uczący wywoływany jest T razy, zbiór treningowy $\mathcal{L}$ składa się z N wektorów, $h_t : \mathcal{L} \mapsto y \in \{+1, -1\}$ oznacza hipotezę. Inicjalizujemy początkowe wartości wag wszystkich wektorów: $D_1(i) = 1/N, \;\; i = 1, \ldots, N$.

Dla $t = 1, \ldots, T$ przeprowadzamy następujące kroki:

- Trenujemy algorytm używając zbioru $\mathcal{D}_t$.
- Testujemy otrzymaną hipotezę obliczając błąd:

$$\epsilon_t = \sum_{i=1}^{N} D_t(i)[1 - \delta(h_i(\mathbf{X}_i), y_i)].$$

- Obliczamy parametr:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right).$$

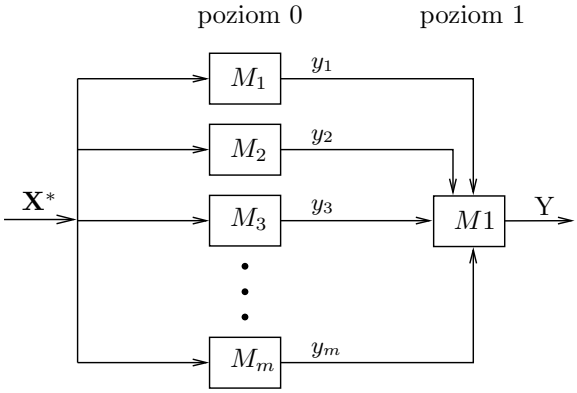
- Uaktualniamy zbiór wag wszystkich N wektorów:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{jeżeli } \; h_i(\mathbf{X}_i) = y_i \\ e^{\alpha_t} & \text{jeżeli } \; h_i(\mathbf{X}_i) \neq y_i \end{cases}$$

  wprowadzając dodatkowo parametr normalizacyjny Z_t.

Końcową hipotezę otrzymujemy z wzoru:

$$\tilde{h}(\mathbf{X}) = \operatorname{sgn} \left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{X}) \right).$$



Rysunek 29: Schemat algorytmu *stacking* składającego się z dwóch poziomów (najpopularniejsza odmiana). W warstwie ostatniej (poziom 1) znajduje się klasyfikator podejmujący końcową decyzję.

W [17] opisane zostają algorytmy *AdaBoost.M1* oraz *AdaBoost.M2*, będące implementacjami wieloklasowymi algorytmu głównego. Zmiany polegają na innym obliczaniu czynnika α_t, sposobie aktualizacji wag D_t oraz zmodyfikowanym obliczaniu końcowej hipotezy.

Arcing

Arcing [3] jest odmianą baggingu, nazwa metody pochodzi od słów ***A****daptively* ***R****esample and* ***C****ombine*. Mając N elementów w zbiorze treningowym, początkowo losujemy wektory z prawdopodobieństwem $p=\frac{1}{N}$. Oznaczamy wektory błędnie klasyfikowane i następne tworzone klasyfikatory pracują już na zbiorach danych, w których prawdopodobieństwo wylosowania wektora błędnego jest większe (w wyniku skalowania prawdopodobieństw $p \mapsto \beta p,\ \ \beta>1$). W ten sposób nowe modele przystosowują się bardziej do obszarów źle klasyfikowanych przez poprzedników. Końcowe prawdopodobieństwo powstaje poprzez proste lub ważone głosowanie.

Stacking

W metodzie tej [37] wprowadzona zostaje architektura warstwowa komitetu (rysunek 29), każda warstwa posiada własny zbiór treningowy. Poziom 0 zawsze bazuje na danych oryginalnych treningowych, każdy z klasyfikatorów poziomu dokonuje własnej klasyfikacji. Na podstawie uzyskanych wyników tworzy się zbiór danych treningowych dla następnego poziomu. Wyniki te przekazywane są do następnej warstwy, która również przekazuje swoje wy-

niki następnej. Ostatnia warstwa (poziom) składa się zawsze z pojedynczego klasyfikatora, który podejmuje końcową decyzję. Podział danych na zbiory przypisane poszczególnym klasyfikatorom może odbywać się na różne sposoby – najpopularniejsza jest odmiana *stackingu* z kroswalidacją. Dokonujemy f-krotnej kroswalidacji zbioru treningowego, dla każdego etapu przeprowadzamy następujące obliczenia:

- Trenujemy wszystkie klasyfikatory poziomu 0.
- Testujemy wszystkie klasyfikatory poziomu 0 na zbiorze testowym powstałym w wyniku kroswalidacji.
- Na podstawie zbioru testowego i wyników testu tworzymy zbiór treningowy dla poziomu 1.

Mając dane dla następnego poziomu można przeprowadzić dla niego powyższe postępowanie i kontynuować proces, aż do osiągnięcia poziomu ostatniego.

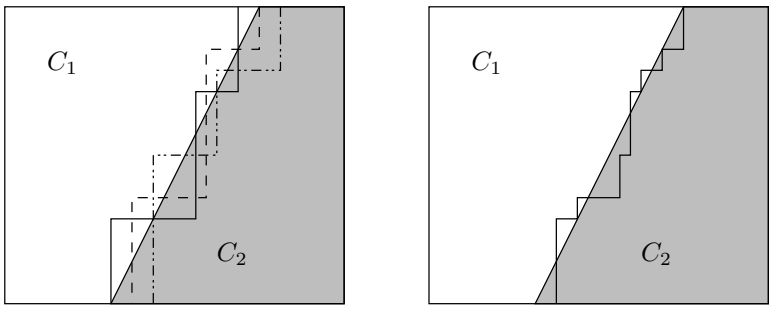
Podejście zastosowane w *stackingu* (wykorzystywanie wewnętrznych zbiorów testowych do tworzenia danych treningowych) szczególnie dobrze wpływa na zmniejszenie błędu generalizacji.

Komitet kroswalidacyjny

Ta bardzo prosta odmiana komitetu opiera się na idei kroswalidacji. Dzielimy zbiór treningowy na ustaloną liczbę (f) podzbiorów rozłącznych. Z powstałych podzbiorów tworzymy f nowych zbiorów treningowych każdorazowo nie włączając do zbioru głównego jednego z podzbiorów (za każdym razem innego). Powstałe zbiory są zbiorami treningowym dla f modeli, których decyzje, po zakończeniu uczenia, są kombinowane.

3.3.2 Komitety oparte na kombinacji głosów lub głosowaniu

W tej grupie komitetów podstawową cechą jest to, iż każdy z członków komitetu pracuje niezależnie na tym samym zbiorze danych (pełny zbiór treningowy). Dla funkcjonowania komitetu interesujące są tylko wyniki klasyfikacji każdego modelu (K prawdopodobieństw $P(C|\mathbf{X})$ lub ostateczna decyzja o przewidywanej klasie). W związku z tym teoretyczne podstawy są tu mniej rozbudowane – nie istnieje pojęcie pracy całego komitetu rozumiane w sensie generowania klasyfikatorów, choć mogą pojawić się elementy wymagające uczenia na zbiorze treningowym.



Rysunek 30: Granica decyzji dla problemu dwuklasowego. Z lewej strony granice decyzji trzech różnych drzew C4.5, z prawej strony granica utworzona w wyniku głosowania większościowego. Obserwujemy lepsze dopasowanie się do rzeczywistego podziału zbioru.

Komitet otrzymuje wyniki od nauczonych klasyfikatorów, podjęcie końcowej decyzji sprowadza się do odpowiedniej analizy zgromadzonych danych. Cała uwaga zostaje więc zwrócona na odpowiednie skonstruowanie optymalnej funkcji decyzyjnej – nie jest natomiast w żaden sposób rozważana optymalna struktura komitetu (modele mogą być dobierane „ręcznie"). W podjęciu decyzji uczestniczą wszyscy członkowie komitetu lub grupa – ich wzajemna współpraca ma dać jak najlepszy wynik.

Takie podejście sprzyja większej niezależności błędów popełnianych przez poszczególne klasyfikatory – brak jest przecież pomiędzy nimi jakiejkolwiek korelacji. Ponadto dużą zaletą jest to, iż do komitetu można włączać klasyfikatory dowolnego rodzaju. Poniżej omówione zostają najczęściej spotykane metody podejmowania decyzji

Głosowanie większościowe

Jest to najbardziej demokratyczny schemat głosowania. Pozwalamy każdemu z członków komitetu podjąć indywidualną decyzję na temat klasyfikacji, następnie zliczamy głosy wszystkich klasyfikatorów. Klasą zwycięską zostaje ta klasa, na którą głosowała największa liczba klasyfikatorów (rysunek 30).

Głosowanie większościowe można zmodyfikować wprowadzając pojęcie progu. Wówczas także zliczamy głosy modeli wchodzących w skład komitetu, jednak pod uwagę bierzemy tylko te głosy, których prawdopodobieństwo

$P(C|\mathbf{X})$ przekroczyło pewną ustaloną wartość. Ten zabieg ma wyeliminować modele, które podejmują decyzję na podstawie słabego przekonania o jej słuszności (mała wartość prawdopodobieństwa).

Ważona kombinacja wyników

Metoda ta może służyć jako podstawa wielu wyrafinowanych algorytmów. Tworzymy ważoną kombinację liniową wyników wielu klasyfikatorów, zgodnie z wzorem:

$$p(C_j|\mathbf{X}^*) = \sum_{l=1}^{m} W_{j,l} P(C_j|\mathbf{X}^*; M_l),$$

gdzie p jest prawdopodobieństwem końcowym komitetu, a P jednego z jego członków. Wektor $\mathbf{X}^*$ przypisujemy do klasy, która uzyskała maksymalne prawdopodobieństwo p. Współczynniki $\underline{\mathbf{W}}$ mogą być wyznaczane na wiele sposobów, wykorzystując informację zawartą w zbiorze treningowym.

Uśrednianie Bayesowskie

Uśredniamy prawdopodobieństwa *a posteriori* otrzymane z pojedynczych klasyfikatorów :

$$p_{av}(C_j|\mathbf{X}^*) = \frac{1}{m} \sum_{l=1}^{m} P(C_j|\mathbf{X}^*; M_l).$$

Końcowej klasyfikacji dokonujemy według kryterium Bayesa, tzn. wybieramy maksymalne prawdopodobieństwo *a posteriori*:

$$\text{Klasa}[\mathbf{X}^*] = \arg \max_{j=1}^{K} [p_{av}(C_j|\mathbf{X}^*)].$$

Belief function

W metodzie tej wykorzystuje się dane o błędach popełnianych na zbiorze treningowym. Wiedza ta zawarta jest w macierzy konfuzji, na jej podstawie obliczyć możemy następujące prawdopodobieństwa:

$$P(\mathbf{X} \in C_i | M_l(\mathbf{X}) = j) = \frac{\mu_{ij}^{(l)}}{\sum_{l=1}^{m} \mu_{ij}^{(l)}},$$

gdzie $\mu_{ij}^{(l)}$ oznacza liczbę wektorów z klasy C_i przypisanych do klasy C_j przez l-ty klasyfikator (tj. $M_l(\mathbf{X}) = j$). Na podstawie powyższych prawdopodo-

bieństw wprowadzona zostaje funkcja:

$$\text{bel}(i) = \eta \prod_{l=1}^{m} P(\mathbf{X} \in C_i | M_l(\mathbf{X}) = j),$$

zawierająca czynnik normalizacyjny η. Wektor zostaje przypisany do klasy, dla której powyższa funkcja osiąga wartość maksymalną.

Nash voting

Metoda bazuje na uśrednianiu Bayesowskim, z tą różnicą, że zamiast średniej głosów obliczamy ich iloczyn:

$$\text{Klasa}[\mathbf{X}^*] = \arg\max_{j=1}^{K} \left[\prod_{l=1}^{m} P(C_j | \mathbf{X}^*; M_l) \right].$$

3.3.3 Mixtures of Experts

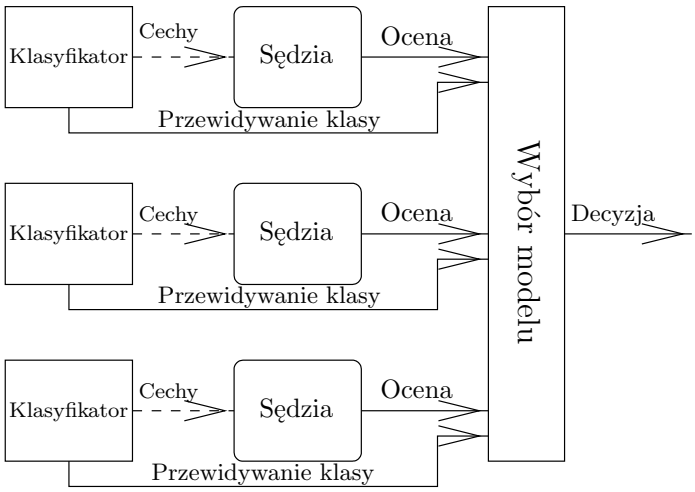
To jedno z najnowszych rozwiązań w dziedzinie komitetów. Zamiast korzystać z grupy klasyfikatorów i na podstawie ich wyników tworzyć końcową decyzję, można starać się zawsze znaleźć jeden, najbardziej kompetentny klasyfikator. Współpracę, występującą w przypadku kombinowania wspólnych wyników (poprzedni podrozdział), zastępuje tu rywalizacja pomiędzy członkami komitetu – dlatego ten rodzaj komitetu często nazywany bywa dynamicznym (ze względu na zmieniające się role poszczególnych klasyfikatorów).

Jeżeli istnieje choć jeden model poprawnie klasyfikujący dany wektor testowy – to on powinien podejmować decyzje. To uwidacznia, iż działanie takiego komitetu powinno się głównie koncentrować na wyborze najbardziej kompetentnego modelu. Dlatego często wprowadza się pojęcie wyroczni, arbitra, sędziego, który wskazuje (po wcześniejszej fazie własnego uczenia się) klasyfikator, któremu należy zawierzyć. Kompetencja danego modelu może być opisywana obszarami w przestrzeni cech lub też po prostu klasami.

Najprostszym ekspertem wydaje się być ten członek komitetu, który osiąga maksymalne prawdopodobieństwo *a posteriori* spośród zbioru wszystkich prawdopodobieństw wszystkich modeli komitetu. Tak stworzony komitet bywa często nazywany *komitetem największego zaufania*:

$$\text{Klasa}[\mathbf{X}^*] = \arg\max_{j=1}^{K} \left[\arg\max_{l=1}^{m} [P(C_j | \mathbf{X}^*; M_l)] \right].$$

Jednak pojedyncze maksymalne prawdopodobieństwo nie jest gwarantem sukcesu z prostego powodu – klasyfikator nie może ocenić sam siebie czy



Rysunek 31: Architektura komitetu *Learned Referees*. Każdy klasyfikator posiada własnego sędziego, który ocenia jego kompetencję. Specjalnie wyodrębniona część zajmuje się wyborem najlepszego modelu.

jego decyzja jest trafna, czy mylna. Stąd dodatkowy „czynnik", mający ocenić kompetencję danego klasyfikatora w konkretnej sytuacji.

Learned Referees

W pracy [32] wprowadzono pojęcie sędziego, przypisanego do każdego modelu wchodzącego w skład komitetu (rysunek 31). Zadanie sędziego to określenie kompetencji modelu w wybranym obszarze przestrzeni cech. Podejście to opiera się na założeniu, iż każdy klasyfikator posiada obszar lub obszary, w których jego klasyfikacja jest najbardziej wiarygodna – sędzia przypisany do modelu ma za zadanie zidentyfikować takie obszary. Końcowa klasyfikacja podejmowana jest przez model, którego „niezawodność" (*reliability*) w danym obszarze jest największa.

Kluczowym zagadnieniem w tym podejściu jest poprawne nauczenie zbioru sędziów (dla każdego klasyfikatora uczenie występuje indywidualnie – można więc używać klasyfikatorów heterogenicznych). Wyznaczenie dobrego sędziego w decydującym stopniu zależy od tego, czy zbiór treningowy, na którym następuje uczenie, jest dobrym reprezentantem całego zbioru danych.

Autorzy zastosowali nowy pomysł wyznaczania obszarów kompetencji każdego z sędziów. W procedurze uczenia wykorzystuje się drzewa decyzyjne

działające na zbiorze danych poetykietowanych na klasy poprawną i niepoprawną (są to jedyne dane wykorzystywane w procesie uczenia komitetu). Po utworzeniu drzewa, dla każdego z liści obliczana jest jego „niezawodność" – otrzymujemy zatem związek pomiędzy konkretnymi cechami (a więc także obszarem w przestrzeni cech) i dokładnością klasyfikacji – na jego podstawie sędzia będzie określał swoją kompetencję.

SCANN

Jest to opisany w [29] rozbudowany model komitetu, którego nazwa wzięła się od trzech wykorzystywanych w nim rozwiązań: ***S**tacking,* ***C**orrespondence* ***A**nalysis,* ***N**earest* ***N**eighbor*. *Stacking* wykorzystywany jest go generowania klasyfikatorów, metoda *CA* modeluje zależność pomiędzy danymi wejściowymi a poprawnością ich klasyfikowania. W ostatniej fazie algorytm najbliższego sąsiada wskazuje najbardziej kompetentny klasyfikator.

Correspondence Analysis jest metodą geometrycznego badania zależności pomiędzy kolumnami i rzędami macierzy, których wartości należą do pewnych kategorii. Zastosowanie jej w *SCANN* ma na celu zbadanie zależności pomiędzy wektorami treningowymi a klasyfikacją przez poszczególne modele. Dane niezbędne do obliczeń są umieszczane zostają w macierzy:

	M_1	M_2	$\dots$	M_m	$\tilde{C}$
$\mathbf{X}_1$	C_{11}	C_{12}	$\dots$	C_{1m}	$\tilde{C}_1$
$\mathbf{X}_2$	C_{21}	C_{22}	$\dots$	C_{2m}	$\tilde{C}_2$
$\mathbf{X}_3$	C_{31}	C_{32}	$\dots$	C_{3m}	$\tilde{C}_3$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{X}_N$	C_{N1}	C_{N2}	$\dots$	C_{Nm}	$\tilde{C}_N$

gdzie C_{Nm} jest klasą przewidywaną przez m-ty model dla N-tego wektora, natomiast $\tilde{C}_a$ jest klasą prawdziwą wektora $\mathbf{X}_a$. Poszukiwanie kierunków głównych, np. przez rozkład *SVD (Singular Value Decomposition)*, prowadzi do dwóch macierzy $\underline{\mathbf{F}}$ i $\underline{\mathbf{G}}$ o liczbie kolumn wynoszącej $I = \min(N-1, m)$. Pierwsza z nich wiąże wszystkie wektory treningowe z poszczególnymi modelami, druga modele z klasami. Na podstawie podobieństwa wierszy z obu macierzy można wyciągnąć ogólną zależność: wiersz $\mathbf{f}_p$ będzie leżał blisko wiersza $\mathbf{g}_r$ wtedy, gdy model m' przewiduje klasę C'.

Do ostatecznego wyboru najlepszego klasyfikatora zostaje zostanie użyty algorytm najbliższego sąsiada (kNN dla $k=1$). Klasy występujące w danych

zostają rzutowane do I-wymiarowej przestrzeni (wykorzystując dane z macierzy $\underline{\mathbf{G}}$), nieznany wektor $\mathbf{X}^*$ zostaje natomiast klasyfikowany przez wszystkie m modeli i na podstawie macierzy $\underline{\mathbf{F}}$ rzutowany w tę samą przestrzeń. Algorytm NN używając metryki euklidesowej oblicza odległość do najbliższej klasy i przypisuje ją nieznanemu wektorowi. Z macierzy $\underline{\mathbf{G}}$ odczytać można który model *de facto* dokonał klasyfikacji.

3.3.4 Pozostałe rodzaje komitetów

Uczenie równoległe

W pracy [28] zaproponowano komitet sieci neuronowych, którego główną cechą jest to, iż uczenie wszystkich sieci następuje równolegle. Istnieje sprzężenie pomiędzy sieciami, parametry (dobierane w procesie uczenia) każdej sieci są uzależnione od pozostałych. Ideą stojąca za takim rozwiązaniem jest założenie, iż każda z sieci świadoma działalności pozostałych członków komitetu może lepiej dopasować się do danych – nie specjalizując się w regionach, gdzie inne modele radzą już sobie dobrze.

Wyjście komitetu obliczane jest jako średnia wszystkich m modeli:

$$Y(i) = \frac{1}{m}\sum_{l=1}^{m} y_l(i),$$

$y_l(i)$ jest wyjściem l-tej sieci w odpowiedzi na $i-$ty wzorzec uczący. Wprowadzamy człon kary do funkcji błędu każdej z sieci tak, że wszystkie sieci mogą być uczone na jednym zbiorze danych. Funkcja błędu l-tej sieci ma postać:

$$E_l = \frac{1}{N}\sum_{i=1}^{N} E_l(i) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y_l(i) - \tilde{y}(i))^2 + \frac{1}{N}\sum_{i=1}^{N}\varrho p_l(i),$$

gdzie ϱ jest stałą mającą wzmocnić lub osłabić $(0 \leqslant \varrho \leqslant 1)$ człon kary $p_l(i)$, wynoszący:

$$p_l(i) = (y_l(i) - Y(i))\sum_{s \not= l}(y_s(i) - Y(i)).$$

Takie podejście ma na celu utworzenie odmiennych od siebie modeli sieci i jednoczesne zwiększenie współpracy sieci w komitecie (poprzez lepsze dopasowanie kompetencji poszczególnych modeli do obszaru danych). W tradycyjnych komitetach złożonych z sieci neuronowych dominuje inne podejście – sieci uczone są niezależnie a ich wyniki później kombinowane. Oczywistym minusem takiego rozwiązania, wynikającym z braku interakcji sieci, jest słabe (nieoptymalne) dopasowanie się do danych wejściowych. Sieci znane są ze

swojej niestabilności i często możliwa jest sytuacja, że wiele modeli nie wyspecjalizuje się w żadnym obszarze, bądź też kilka z nich będzie dobrze „opisywało" ten sam region. Sprzężenie sieci ma zapobiec temu zjawisku. Głównym minusem takiego rozwiązania jest czasochłonny proces uczenia, który ma dodatkowo większe szanse na utknięcie w minimum lokalnym (większa liczba parametrów adaptacyjnych).

Manipulowanie cechami

Metoda ta [9] opiera się na dostarczaniu do algorytmów uczenia poszczególnych klasyfikatorów, różnych zestawów cech danych wejściowych. Ponieważ zbiór cech tworzy przestrzeń, ma to uzasadnienie, gdyż różne modele mogą się dostosować do wybranych podprzestrzeni w mniejszym lub większym stopniu. Metodę stosuje się głównie dla zbiorów danych posiadających wiele cech, prawdopodobnie niektóre z nich mogą być zbędne.

Randomizacja

Randomizacja [9] polega na wykorzystaniu elementów losowych pojawiających się w procesie uczenia klasyfikatora (dlatego najłatwiej wprowadzić ją można do sieci neuronowych). Jeżeli ten sam algorytm uczenia wykorzystujący losowe wagi początkowe wykorzystujemy w wielu sieciach, otrzymane klasyfikatory będą zupełnie różne. Randomizację zastosowano również do drzew decyzji. Losowość opiera się tam na tym, iż decyzja, którą cechę analizować na którym węźle, jest przypadkowa. Badania wykazały, że randomizowane drzewa wypadały lepiej lub na tym samym poziomie co pojedyncze drzewo.

4 Model CUC

4.1 Idea modelu

W większości typowych komitetów końcowe decyzje podejmowane są zgodnie z procedurami demokratycznymi, nie ma modeli całkowicie dyskryminowanych ani uprzywilejowanych. Przez wiele lat było to zasadniczą cechą komitetów. Z czasem podejście to okazało się jednak nie być najlepszym rozwiązaniem. Już z codziennego życia wiadomo, iż nie ma specjalistów od wszystkiego, patrząc choćby na rozwój współczesnej nauki, zauważamy w niej bardzo wąskie specjalizacje, a czasy mędrców obejmujących swoją wiedzą całą dziedzinę odeszły na zawsze do lamusa. Także w problemach, z którymi boryka się Inteligencja Obliczeniowa nie ma narzędzi uniwersalnych. Wśród istniejących klasyfikatorów zauważamy zróżnicowanie pod względem zastosowań w konkretnych przypadkach, dobry wynik na jednym zbiorze danych nie gwarantuje sukcesu na następnym. Często dany model ma zastosowanie tylko do pewnego podzbioru analizowanych danych. Sytuacja taka nie może jednak być powodem do odrzucania nieobiecujących klasyfikatorów, dobrze stworzony komitet powinien umieć wydobyć ze zbioru modeli maksymalnie dużo takich informacji, by końcowa klasyfikacja była jak najlepsza.

Model niedemokratyczny *Competent Undemocratic Committee* (*CUC*) powstał jako narzędzie mające stworzyć optymalną kombinację liniową wybranych klasyfikatorów. „Niedemokratyczność" wpisana w komitet polega na tym, iż możliwe są sytuacje, gdy decyzja podejmowana jest przez jeden model – jego wkład do kombinacji liniowej będzie największy. W modelu *CUC* każdy klasyfikator analizowany jest pod względem zastosowania w pewnym (większym lub mniejszym) podzbiorze danych, model który osiąga ogólną – bardzo słabą klasyfikację, nie jest dyskwalifikowany, może się on okazać bardzo przydatny tam, gdzie pozostałe, działające dobrze klasyfikatory wypadają słabo. Naturalną sytuacją jest, że wektory klasyfikowane błędnie przez niektóre modele mogą być poprawnie klasyfikowane przez pozostałe. Jeżeli choć jeden z modeli wchodzących w skład komitetu klasyfikuje dany wektor poprawnie, istnieje szansa na poprawną klasyfikację danego wektora przez komitet. Model *CUC* nie wyznacza jednak globalnych obszarów lepszej lub gorszej klasyfikacji, dla każdego z członków komitetu poszukujemy lokalnych miejsc (nazywanym obszarami niekompetencji) słabej klasyfikacji, przypisujemy obszary do poszczególnych modeli i wykluczamy odpowiednie modele (zmniejszając rangę ich głosu), gdy przypadek poddawany klasyfikacji znajdzie się w obszarze przypisanym do danego klasyfikatora. Obszary niekompetencji mogą być utożsamiane z klasami, lecz nie muszą.

Niech prawdopodobieństwo przypisania wektora $\mathbf{X}$ do klasy C_j przez model M_l oznaczone będzie jako $P(C_j|\mathbf{X}; M_l)$, indeksy zmieniają się w granicach: $j = 1, 2, \ldots, K$, $l = 1, 2, \ldots, m$. Podjęcie decyzji przez komitet to ważona suma głosów pojedynczych modeli, należących do zbioru $\mathcal{M}$:

$$p(C_j|\mathbf{X}; \mathcal{M}) = \sum_{l=1}^{m} W_{j,l} P(C_j|\mathbf{X}; M_l). \tag{6}$$

Otrzymujemy więc $m \cdot K$ dodatkowych parametrów, tworzących macierz $\underline{\mathbf{W}}$, które należy obliczyć. Do wskazywania obszarów niekompetencji służyć będą wektory referencyjne $\mathbf{R}$ przypisane do konkretnego, l-tego modelu, tworzące zbiór $\mathcal{R}_l$. Wyznaczanie ich prezentuje poniższy algorytm:

1. Uczenie wszystkich m modeli na zbiorze treningowym $\mathcal{L}$.
2. Dla każdego modelu M_l:
   (a) wyznaczanie klasy przewidywanej $C_l(\mathbf{X})$ dla wszystkich wektorów treningowych;
   (b) jeśli $C_l(\mathbf{X}) \neq C(\mathbf{X})$, tj. model M_l błędnie klasyfikuje wektor $\mathbf{X}$, należy wyznaczyć obszar niekompetencji danego modelu w otoczeniu wektora $\mathbf{X}$ obliczając odległość d do najbliżej położonego wektora klasyfikowanego poprawnie przez model M_l. Obszar ten będzie opisywany wektorem $\mathbf{R}$ dodanym do zbioru $\mathcal{R}_l$;
   (c) ustalenie parametrów funkcji niekompetencji $F(\|\mathbf{X}^* - \mathbf{R}\|)$ tak, by jej wartość malała znacząco dla $\|\mathbf{X}^* - \mathbf{R}\| \leqslant d$.
3. Funkcja niekompetencji dla modelu $F(\mathbf{X}^*; M_l)$ jest iloczynem funkcji $F(\|\mathbf{X}^* - \mathbf{R}\|)$ dla wszystkich wektorów $\mathbf{R}$ ze zbioru $\mathcal{R}_l$:

$$F(\mathbf{X}^*; M_l) = \prod_{\mathbf{R} \in \mathcal{R}_l} F(\|\mathbf{X}^* - \mathbf{R}\|). \tag{7}$$

Funkcja F powinna spełniać warunki:

- We wszystkich obszarach, gdzie model M_l poprawnie klasyfikuje:

$$F(\mathbf{X}^*; M_l) = 1.$$

- W obszarach słabej klasyfikacji:

$$F(\mathbf{X}^*; M_l) \approx 0.$$

W roli F występować może wiele funkcji, których rozmiary można uzależniać od wielkości d[16]:

- Funkcja oparta na funkcji gaussowskiej:

$$F(\|\mathbf{X}^*-\mathbf{R}\|)=1-G(\|\mathbf{X}^*-\mathbf{R}\|^a)=1-\exp\left(-\frac{\|\mathbf{X}^*-\mathbf{R}\|^a}{2\sigma^2}\right). \tag{8}$$

- Funkcja oparta na uproszczonej funkcji wykładniczej ($e^x \approx 1+x$):

$$F(\|\mathbf{X}^*-\mathbf{R}\|)=\left[1+\left(\frac{\|\mathbf{X}^*-\mathbf{R}\|^{-a}}{2\sigma^2}\right)\right]^{-1}. \tag{9}$$

- Funkcja oparta na funkcji sigmoidalnej:

$$F(\|\mathbf{X}^*-\mathbf{R}\|)=1-\frac{1}{1+\exp\left[\theta(\|\mathbf{X}^*-\mathbf{R}\|-d)\right]}. \tag{10}$$

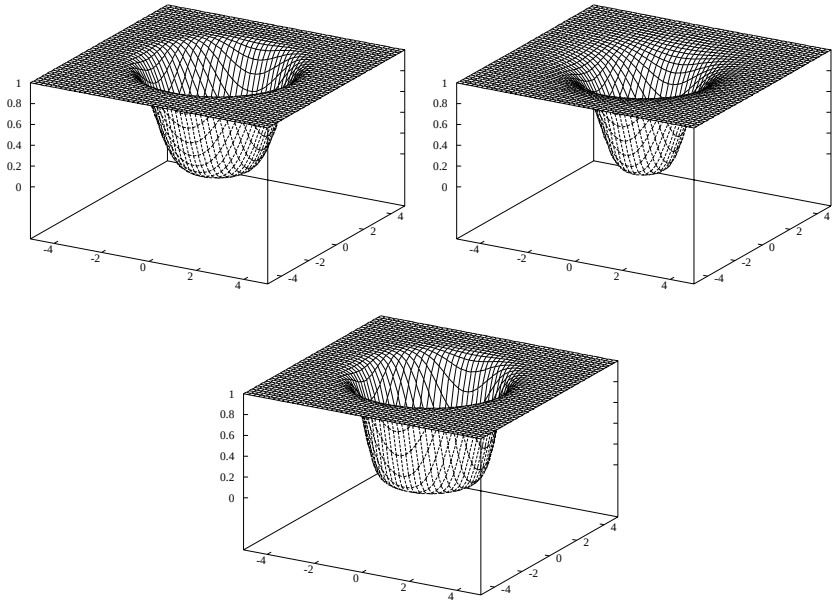
Parametr $a \geqslant 1$ wprowadzono w celu regulacji spłaszczenia funkcji. Wszystkie powyższe funkcje przedstawione są na rysunku 32. Ponieważ funkcja F powinna osiągać wartość 1 możliwie szybko poza obszarem niekompetencji, używać należy dużych wartości a lub skosów sigmoid θ oraz wprowadzać wartości graniczne, po przekroczeniu których funkcja przyjmie wartość „1”[17].

Współczynniki $\underline{\mathbf{W}}$ pomnożone przez funkcje niekompetencji modyfikują wartości prawdopodobieństw otrzymanych z poszczególnych modeli tak, że gdy poddawany klasyfikacji wektor $\mathbf{X}^*$ znajdzie się wewnątrz obszaru referencyjnego danego modelu, waga głosu tego modelu w komitecie zostaje zmniejszona. Uwidacznia się to przeskalowaniem l-tej kolumny macierzy $\underline{\mathbf{W}}$, czyli zmniejszeniem K współczynników $W_{j,l}$ występujących w kombinacji liniowej. Dla klasy C_j ostateczne prawdopodobieństwo klasyfikacji otrzymujemy modyfikując wzór (6):

$$p(C_j|\mathbf{X}^*;\mathcal{M}) = \sum_{l=1}^{m} W_{j,l} F(\mathbf{X}^*;M_l) P(C_j|\mathbf{X}^*;M_l). \tag{11}$$

[16] W przypadku dwóch pierwszych funkcji zależność $F(d)$ otrzymujemy poprzez związek $F(\sigma(d))$.

[17] Ma to także uzasadnienie z numerycznego punktu widzenia. Mnożąc dowolną liczbę przez kolejne czynniki $\lesssim 1$ można ją, przy dostatecznie dużej liczbie mnożeń, sprowadzić do wartości bliskiej 0, co byłoby tu efektem niepożądanym, gdyż prowadziłoby do szybkiego „wyzerowania” wszystkich współczynników macierzy $\underline{\mathbf{W}}$.



Rysunek 32: Funkcje F dla $d=3$. U góry funkcje (8) oraz (9) dla $a=5$. Niżej funkcja (10) dla $\theta=5$. Wszędzie przyjęto $\mathbf{R}=\vec{0}$.

Podstawiając $\underline{\mathbf{W}}'(\mathbf{X}^*) \mapsto \underline{\mathbf{W}}F(\mathbf{X}^*; M_l)$ zauważamy, iż nowo powstałe współczynniki $\underline{\mathbf{W}}'$ zależą bezpośrednio od klasyfikowanego wektora $\mathbf{X}^*$, nie jest to zatem „statyczna" kombinacja liniowa podobna do występującej we wzorze (6), lecz dynamiczna, dostosowująca swoją strukturę do każdego nowego przypadku.

W powyższym rozwiązaniu można doszukiwać się analogii (a może raczej inspiracji) neurobiologicznych. Aktywność neuronów w mózgu (wyrażana przez wielkości wag $\underline{\mathbf{W}}$), tworzących wyspecjalizowane obszary, może przybierać różny poziom, wysoki – gdy na podstawie danego pobudzenia można wyciągnąć interesujące wnioski, niski przy „obojętnym" sygnale (w przypadku braku wartościowania sygnałów tak samo reagowalibyśmy na dochodzące do nas sygnały, np. twarze mijanych osób). Uzależniając wagi (tj. parametry decydujące o działaniu całego układu) od sygnałów wejściowych ($\underline{\mathbf{W}} = \underline{\mathbf{W}}(\mathbf{X})$) dajemy komitetowi większą możliwość dopasowania się do konkretnych danych wejściowych (czego nie obserwujemy np. w sieciach neuronowych).

4.2 Program

Komitet *CUC* analizuje prawdopodobieństwa $P(C_j|\mathbf{X}; M_l)$ i tworzy z nich optymalną liniową kombinację opisaną wzorem (11). Niezależnie od pracy każdego klasyfikatora, w obliczeniach komitetu wykorzystywanych jest pięć liczbowych zbiorów danych:

- Zbiór wektorów treningowych $\mathcal{L}$ – wykorzystywany do obliczenia prawdopodobieństw treningowych oraz do wyznaczania wektorów $\mathbf{R}$.
- Zbiór wektorów testowych $\mathcal{T}$ – wykorzystywany do obliczania prawdopodobieństw testowych oraz obliczania odległości $\|\mathbf{X}_T - \mathbf{R}\|$, gdzie $\mathbf{X}_T$ oznacza wektor testowy.
- Zbiór prawdopodobieństw treningowych $\mathcal{P}_L$ o elementach P_L. Indeks L oznacza, iż prawdopodobieństwo $P_L(C_j|\mathbf{X})$ dotyczy wektora treningowego.
- Zbiór prawdopodobieństw testowych $\mathcal{P}_T$ – analogiczny do poprzedniego zbiór dla wektorów testowych.
- Zbiór $\mathcal{R}$ – tworzony dynamicznie w trakcie obliczeń, składa się z wektorów $\mathbf{R}$, pogrupowanych według przynależności do poszczególnych modeli, tj. tworzących podzbiory $\mathcal{R}_l$.

Zbiory treningowe służą do ustalania parametrów komitetu (współczynników $\underline{\mathbf{W}}$, zbiorów $\mathcal{R}_l$, wielkości d) w procesie uczenia, natomiast zbiory testowe są wykorzystywane podczas „pracy" komitetu (tj. klasyfikacji na zbiorze $\mathcal{T}$) poprzez obliczanie współczynników $\underline{\mathbf{W}}'$.

Do obliczania prawdopodobieństw (11) będących decyzją komitetu niezbędna jest znajomość prawdopodobieństw klasyfikacji poszczególnych N wektorów (dla każdego wektora potrzebujemy prawdopodobieństwa przynależności do K klas), przedstawionych w macierzy:

$$\begin{pmatrix} P(C_1|\mathbf{X}_1) & P(C_2|\mathbf{X}_1) & \dots & P(C_K|\mathbf{X}_1) \\ P(C_1|\mathbf{X}_2) & P(C_2|\mathbf{X}_2) & \dots & P(C_K|\mathbf{X}_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(C_1|\mathbf{X}_N) & P(C_2|\mathbf{X}_N) & \dots & P(C_K|\mathbf{X}_N) \end{pmatrix}$$

dodatkowo prawdopodobieństwa takie wyznaczamy dla każdego modelu, oddzielnie dla zbiorów $\mathcal{L}$ i $\mathcal{T}$.

Numeryczna strona wyznaczania współczynników $\underline{\mathbf{W}}'$ wygląda następująco:

1. Obliczanie współczynników bazowych $\underline{\mathbf{W}}$.

   Współczynniki $\underline{\mathbf{W}}$ wyznaczamy z równania (6), zapisując je w postaci:

   $$P_j(\mathbf{X}) = \sum_{l=1}^{m} W_{j,l} P_L(C_j|\mathbf{X}; M_l), \tag{12}$$

   gdzie $P_j(\mathbf{X})$ oznacza „prawdziwe" prawdopodobieństwo, iż wektor treningowy $\mathbf{X}$ należy do klasy C_j, wynosi 1 gdy klasą wektora $\mathbf{X}$ jest C_j i 0 w przeciwnym wypadku (ponieważ operujemy na zbiorze treningowym $\mathcal{P}_L$, można z tej informacji skorzystać). Rozwinięcie elementów równania (12) następuje ze względu na dwa indeksy: liczbę wektorów (wiersze) oraz liczbę klas (kolumny). Rozwinięcie dla N wektorów oraz dla klasy pierwszej przedstawia poniższy układ równań (dla ułatwienia założono uporządkowanie wektorów według klas, ustalono liczbę wektorów w klasie pierwszej jako s oraz przyjęto, iż klasy wektorów treningowych są rozłączne, tzn. prawdopodobieństwa przynależności mogą przyjmować tylko wartości 0 i 1):

   $$\begin{array}{ll}
   1 = W_{11}P_L(1|1;1) & + W_{12}P_L(1|1;2) + \ldots + W_{1l}P_L(1|1;l) \\
   1 = W_{11}P_L(1|2;1) & + W_{12}P_L(1|2;2) + \ldots + W_{1l}P_L(1|2;l) \\
   \quad\vdots & \\
   1 = W_{11}P_L(1|s;1) & + W_{12}P_L(1|s;2) + \ldots + W_{1l}P_L(1|s;l) \\
   0 = W_{11}P_L(1|s+1;1) & + W_{12}P_L(1|s+1;2) + \ldots + W_{1l}P_L(1|s+1;l) \\
   0 = W_{11}P_L(1|s+2;1) & + W_{12}P_L(1|s+2;2) + \ldots + W_{1l}P_L(1|s+2;l) \\
   \quad\vdots & \\
   0 = W_{11}P_L(1|N-1;1) & + W_{12}P_L(1|N-1;2) + \ldots + W_{1l}P_L(1|N-1;l) \\
   0 = W_{11}P_L(1|N;1) & + W_{12}P_L(1|N;2) + \ldots + W_{1l}P_L(1|N;l)
   \end{array}$$

   Skrócony zapis $P(a|b;c)$ oznacza $P(C_a|\mathbf{X}_b; M_c)$, natomiast zmiana wartości $P_j(\mathbf{X})$ z 1 na 0 wyznacza zmianę klasy z pierwszej na drugą. $K-1$ pozostałych układów konstruujemy analogicznie stosując rozwinięcia równania (12) dla $j = 2, 3, \ldots, K$.

   Otrzymujemy K nadokreślonych[18] układów równań, rozwiązaniem[19]

[18] Przy założeniu, że liczba wektorów treningowych jest większa od liczby modeli, co praktycznie jest zawsze spełnione.

[19] Rozwiązanie przedstawione jest w dodatku A.

każdego z nich jest wektor $\mathbf{W}_j = (W_{j1}, W_{j2}, \ldots, W_{jm})$, który stanowi j-ty wiersz macierzy $\underline{\mathbf{W}}$:

$$\underline{\mathbf{W}} = \begin{pmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_K \end{pmatrix} = \begin{pmatrix} W_{11} & W_{12} & \ldots & W_{1m} \\ W_{21} & W_{22} & \ldots & W_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ W_{K1} & W_{K2} & \ldots & W_{Km} \end{pmatrix}$$

2. Wyznaczanie wektorów referencyjnych.

   Wektory referencyjne (a dokładniej punkty w przestrzeni cech wskazywane przez wektory zaczepione w początku układu współrzędnych) wyznaczają obszary słabej klasyfikacji poszczególnych modeli. Każdy model posiada własny zbiór wektorów referencyjnych oznaczony jako $\mathcal{R}_l$, gdzie $l = 1, 2, \ldots, m$. Zatem oprócz niezbędnej informacji o położeniu punktu, wektor $\mathbf{R}$ musi również zawierać informację o numerze modelu, którego dotyczy oraz o rozmiarach wyznaczanego obszaru.

   Prototypami do wyznaczenia wektorów referencyjnych są wektory treningowe, które były źle klasyfikowane przez poszczególne modele. Wybieramy jednak tylko takie błędne wektory, które były poprawnie klasyfikowane przez przynajmniej jeden z modeli. Przykładowe wektory zawiera poniższa tabelka (kolumny numerują modele, w wierszach znajdują się wektory, „+" oznacza prawidłową klasyfikację, „−" błąd):

	M_1	M_2	M_3	M_4	$\ldots$
$\mathbf{X}_1$	+	−	+	−	
$\mathbf{X}_2$	−	−	−	+	
$\mathbf{X}_3$	+	+	+	+	
$\mathbf{X}_4$	−	−	−	−	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

   Widzimy, że wektor $\mathbf{X}_1$ może służyć jako wektor referencyjny dla modeli M_2 i M_4, natomiast $\mathbf{X}_2$ dla modeli M_1, M_2 i M_3. Wektory $\mathbf{X}_3$ oraz $\mathbf{X}_4$ nie będą należały do żadnego zbioru $\mathcal{R}$, drugi z nich może zastać uznany jako niedający się sklasyfikować. Aby wyznaczyć granicę obszarów niekompetencji d, należy zbadać sąsiedztwo źle klasyfikowanych wektorów, przy okazji można zmniejszyć liczbę wektorów referencyjnych, gdy kilka z nich leży blisko siebie i nie ma pomiędzy nimi wektorów klasyfikowanych prawidłowo. Wektor referencyjny $\mathbf{R}$ pochodzący od wektora $\mathbf{X}_1$ jest tworzony następująco:

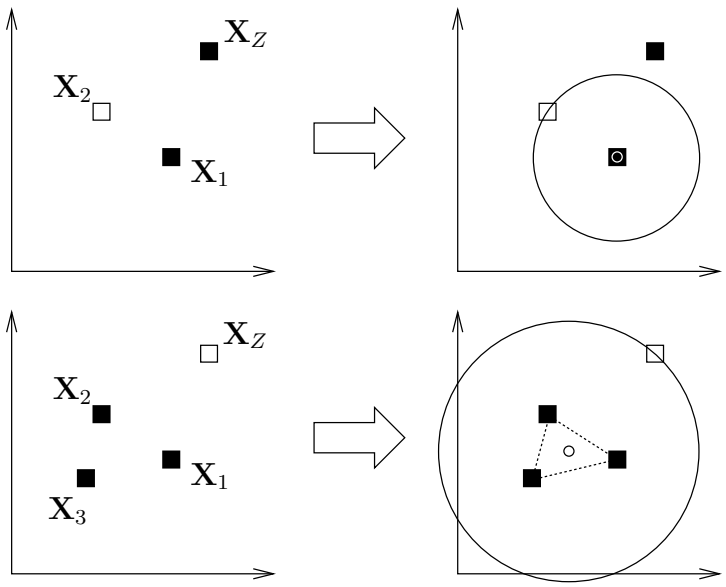
- Ustalamy $\mathbf{R}=\mathbf{X}_1$ i zaczynamy poszukiwania najbliższego poprawnie klasyfikowanego (przez dany model) wektora ze zbioru $\mathcal{L}$.
- Jeżeli najbliższy znaleziony sąsiad wektora $\mathbf{X}_1$ – wektor $\mathbf{X}_2$ jest jednak także źle klasyfikowany, zmieniamy położenie wektora $\mathbf{R}$ zgodnie z wzorem:
$$\mathbf{R}' = \frac{\mathbf{R}+\mathbf{X}_2}{2}, \tag{13}$$
gdzie $\mathbf{R}'$ oznacza nowe położenie wektora $\mathbf{R}$. Wektor $\mathbf{X}_2$ nie będzie już prototypem wektora referencyjnego dla danego modelu w dalszym postępowaniu.
- Poszukujemy kolejnych, coraz dalszych najbliższych sąsiadów, tak długo (gdy liczba sąsiednich wektorów źle klasyfikowanych jest większa od jeden ($\gamma > 1$), położenie wyznaczamy zgodnie z wzorem na geometryczny środek punktów: $\mathbf{R}' = \frac{\mathbf{R}+\sum_\gamma \mathbf{X}}{\gamma+1}$), aż znajdziemy wektor poprawnie sklasyfikowany $\mathbf{X}_Z$, wówczas ustalamy parametr odpowiedzialny za rozmiar obszaru:
$$d = \|\mathbf{X}_Z - \mathbf{R}\|.$$

Procedura ta przedstawiona jest na rysunku 33.

Mając wyznaczone d, należy ustalić takie parametry funkcji F, by dostosować jej rozmiary do opisywanego przez nią obszaru. Należy w każdym przypadku ustalić dwie wielkości funkcji: wielkość odpowiedzialną za rozpiętość przestrzenną (zależną od d), drugą regulującą stromość zboczy funkcji (wartość niezależna). Niech λ oznacza tę wartość graniczną funkcji F (zatem $0 < \lambda < 1$), że po spełnieniu warunku $F \geqslant d$ nastąpi „przycięcie" funkcji i zastąpienie jej wartości wartością 1. Żądamy, aby w punkcie przycięcia d funkcja F przyjmowała konkretną wartość λ, zatem dla poszczególnych funkcji otrzymujemy następujące wzory na parametry regulujące kształt funkcji (wszędzie dla uproszczenia przyjęto $\mathbf{R}=\vec{0}$):

- Funkcja (8):

  Parametrem rozpiętości funkcji jest σ, otrzymujemy:
$$F(d)=\lambda=1-\exp\left(-\frac{d^a}{2\sigma^2}\right) \quad \Rightarrow \quad \sigma(\lambda,d)=\sqrt{-\frac{d^a}{2\ln(1-\lambda)}}. \tag{14}$$



Rysunek 33: Dwa przypadki wyznaczania obszarów referencyjnych, „○" symbolizuje położenie centrum – punkt wskazywany przez wektor $\mathbf{R}$, „□" oznacza poprawną klasyfikację, „■" błąd. U góry obserwujemy najbliższego sąsiada $\mathbf{X}_1$ jako wektor klasyfikowany prawidłowo. Na dole: najbliższy wektor jest źle klasyfikowany – obserwujemy przemieszczenie centrum, trzy prototypy wchodzą w skład jednego obszaru. Promień d okręgu wyznacza granice obszaru referencyjnego.

- Funkcja (9):

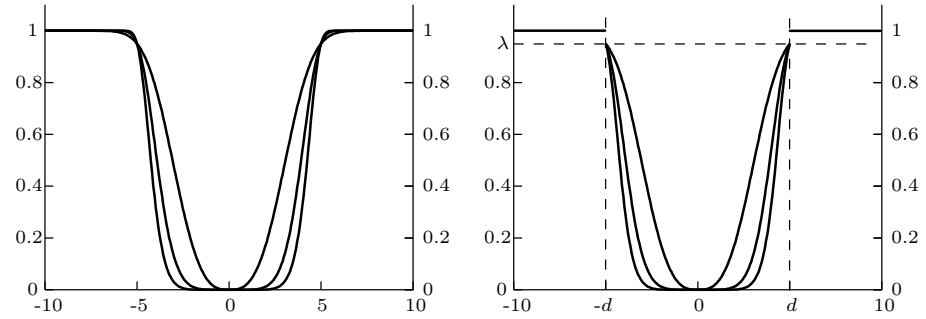
  Analogicznie jak w poprzedniej funkcji:

$$F(d)=\lambda=\left[1+\left(\frac{d^{-a}}{2\sigma^2}\right)\right]^{-1} \quad \Rightarrow \quad \sigma(\lambda,d)=\sqrt{\frac{\lambda d^{-a}}{2(1-\lambda)}}. \qquad (15)$$

- Funkcja (10):

  Parametrem rozpiętości jest d, jednak, ponieważ w punkcie tym funkcja przyjmuje zawsze wartość 0.5, należy przeskalować wielkość d tak, by dla nowej wielkości d' wartość funkcji w d wynosiła λ:

$$F(d)=\lambda=1-\frac{1}{1+\exp\left[\theta(d-d')\right]} \quad \Rightarrow \quad d'(\lambda,d)=\frac{\theta d-\ln\left(\frac{\lambda}{1-\lambda}\right)}{\theta}. \qquad (16)$$



Rysunek 34: Z lewej strony wykres funkcji (8) dla trzech wartości $a: 3, 6, 9$ (im większa wartość a, tym krzywa bardziej dąży do funkcji prostokątnej), przeskalowanej przy pomocy wzoru (14) dla $d=5$, $\lambda=0.95$. Z prawej strony ilustracja przycinania omawianej funkcji w punkcie d o wartości λ.

Parametr λ w zasadzie może być dowolny, jednak sensowne wydają się być tylko wartości ≈ 1. Zabieg przycinania jednej z funkcji ilustruje rysunek 34.

3. Modyfikacja współczynników $\underline{\mathbf{W}}$.

   Modyfikacja współczynników $\underline{\mathbf{W}}$, czyli ustalanie optymalnej kombinacji liniowej wyrażanej wzorem (11), to podstawa działania komitetu. Mając nieznany przypadek $\mathbf{X}^*$ oraz l zbiorów $\mathcal{R}_l$ z wyznaczonymi wektorami, należy sprawdzić w którym obszarze może potencjalnie znajdować się wektor $\mathbf{X}^*$. Przeszukiwanie odbywa się po zbiorach $\mathcal{R}_l$, gdy dany wektor znajdzie się w odpowiednim obszarze (opisywanym wektorem $\mathbf{R}$) zostaje zmieniona cała kolumna macierzy $\underline{\mathbf{W}}$ związana z tym modelem. Zmiana elementów macierzy polega na pomnożeniu ich przez wartość funkcji $F(\|\mathbf{X}^* - \mathbf{R}\|)$. Ponieważ dany wektor może znajdować się w kilku obszarach wyznaczanych przez różne wektory $\mathbf{R}$, wybrane elementy macierzy $\underline{\mathbf{W}}$ mogą być pomniejszane wielokrotnie (zgodnie z wzorem (7)).

Mając wyznaczoną macierz $\underline{\mathbf{W}}'$ dla wektora $\mathbf{X}^*$ można dokonać końcowej klasyfikacji używając prawdopodobieństw testowych P_T. Z wzoru (11) wyz-

naczamy:

$$\begin{aligned} p(C_1|\mathbf{X}^*;\mathcal{M}) &= \sum_{l=1}^{m} W'_{1,l} P_T(C_1|\mathbf{X}^*;M_l) \\ p(C_2|\mathbf{X}^*;\mathcal{M}) &= \sum_{l=1}^{m} W'_{2,l} P_T(C_2|\mathbf{X}^*;M_l) \\ &\vdots \\ p(C_K|\mathbf{X}^*;\mathcal{M}) &= \sum_{l=1}^{m} W'_{K,l} P_T(C_K|\mathbf{X}^*;M_l). \end{aligned} \tag{17}$$

By spełnić warunek:

$$\sum_{j=1}^{K} p(C_j|\mathbf{X}^*;\mathcal{M}) = 1,$$

można przeskalować wyniki (17) transformacją *softmax*:

$$p'(C_j|\mathbf{X}^*;\mathcal{M}) = \frac{\exp\left(p(C_j|\mathbf{X}^*;\mathcal{M})\right)}{\sum_{q=1}^{K} \exp(p(C_q|\mathbf{X}^*;\mathcal{M}))}.$$

Ostateczną przynależność wektora $\mathbf{X}^*$ do klasy C_j stwierdzamy na podstawie wzoru:

$$\text{Klasa}[\mathbf{X}^*] = \arg\max_j \left[p'(C_j|\mathbf{X}^*;\mathcal{M})\right], \quad j = 1, 2, \ldots, K.$$

4.3 Otrzymane wyniki

Numeryczną implementację komitetu *CUC* przetestowano na kilku dostępnych zbiorach danych, w trakcie obliczeń przestrzegano ścisłego podziału na część treningową ($\mathcal{L}$) i testową ($\mathcal{T}$). Do wygenerowania modeli wchodzących w skład komitetu użyto specjalnej wersji programu GhostMiner 1.0 oraz programu *SBL* (obydwa powstały w Katedrze Informatyki Stosowanej).

W czasie doświadczeń numerycznych przetestowano omawiane wcześniej typy klasyfikatorów z rożnymi parametrami, jednak nie wszystkie z nich zostały włączone do komitetu. W przypadku większych zbiorów danych sprawdzenie kilku klasyfikatorów okazało się niemożliwe albo ze względu na czas obliczeń, albo na wymagania pamięciowe. Głównym celem eksperymentów numerycznych nie było jednak stworzenie jak najlepszych pojedynczych klasyfikatorów, ale wygenerowanie takich modeli, by ich wzajemna współpraca przełożyła się na optymalne, całościowe działanie komitetu.

Członków komitetu dobierano eksperymentalnie, kierując się różnorodnością badaną poprzez analizę macierzy konfuzji. Liczba modeli składowych i jej wpływ na końcową klasyfikację nie były szczegółowo rozpatrywane, mogą być przedmiotem dalszej analizy.

Wszystkie przedstawione wyniki uzyskano używając jako funkcji niekompetencji zmodyfikowanej funkcji Gaussa (8), gdyż od niej zaczynano badania. Inne funkcje dawały wyniki takie same, bądź nieznacznie gorsze. Pozostałe parametry komitetu zostają podane przy konkretnych wynikach.

Dane klasyfikacyjne podawane są w procentach i prezentują dokładność klasyfikacji.

Dla każdego zbioru danych zamieszczono następujące informacje i wyniki:

- Ogólne informacje o zbiorze (liczba wektorów, klas, cech), liczebność wektorów w poszczególnych klasach.
- Najlepsze pojedyncze klasyfikatory – wyniki[20] najlepszych klasyfikatorów na danym zbiorze, jego części treningowej i testowej.
- Modele wybrane do komitetu – opis modeli wchodzących w skład komitetu *CUC*.

[20]Pochodzą one ze strony www.phys.uni.torun.pl/kmk/projects/datasets.html.

W przypadku braku podanych parametrów, stosowano domyślne ustawienia programów.

- Klasyfikacje za pomocą wybranych modeli w poszczególnych klasach dla zbioru treningowego i testowego – prezentują zróżnicowanie klasyfikacji członków komitetu w różnych klasach.

- Najlepszą możliwą klasyfikację dla wybranych modeli – najlepsza klasyfikacja, jaką może osiągnąć komitet na podstawie zabranych wyników pojedynczych klasyfikatorów tworzących komitet. Jest po wielkość teoretyczna, powstała poprzez analizę wszystkich wektorów i zakwalifikowaniu danego wektora jako klasyfikowanego poprawnie, gdy przynajmniej jeden z modeli bezbłędnie przewiduje jego klasę.

- Otrzymaną macierz współczynników bazowych $\underline{\mathbf{W}}$. Jest to macierz niezmodyfikowanych współczynników z równania (12), obliczona na podstawie danych treningowych.

- Wynik komitetu CUC na zbiorach $\mathcal{L}$ i $\mathcal{T}$ wraz z porównaniami. Ponieważ twórcy komitetów rzadko testują swoje pomysły na ogólnodostępnych zbiorach danych, zastosowano porównanie z komitetem opartym na głosowaniu większościowym oraz z komitetem największego zaufania, obydwa można w łatwy sposób stworzyć samemu. Dodatkowo zamieszczono parametry komitetu CUC prowadzące do końcowej klasyfikacji.

- Przedział ufności dla otrzymanego wyniku (końcowej klasyfikacji komitetu CUC na zbiorze testowym). Przedział ufności określa zakres wartości wokół zmiennej, w którym, przy danym poziomie pewności ($\bar{p}$), oczekujemy „prawdziwej" wartości zmiennej. Innymi słowy, nawet gdy testowana hipoteza jest słuszna, prawdopodobieństwo przybrania przez zmienną wartości z poza przedziału wynosi $1 - \bar{p}$.

  Obliczając przedział ufności opieramy się na dwustronnym t teście weryfikacji hipotez [2]. Test ten pozwala nam odpowiedzieć na pytanie kiedy dwa klasyfikatory, popełniające błędy, można uznać za statystycznie rozróżnialne z określonym prawdopodobieństwem. Rozważmy klasyfikatory A i B, błędnie klasyfikujące odpowiednio N_A oraz N_B wektorów. Zakładając, że poddawany klasyfikacji przez model A wektor $\mathbf{X}$ ze zbioru $\mathcal{T}$ będzie niepoprawnie klasyfikowany z prawdopodobieństwem:

$$P_A = \frac{N_A}{N},$$

liczba błędnych klasyfikacji w całym zbiorze będzie opisywana dwumianowym rozkładem prawdopodobieństwa (5), który dla dostatecznie dużego N można przybliżyć rozkładem normalnym (analogicznie w przypadku klasyfikatora B). Korzystając z faktu, iż różnica dwóch niezależnych (brak korelacji) zmiennych opisywanych rozkładem normalnym, podlega tej samej statystyce, otrzymujemy zmienną o rozkładzie (w przybliżeniu) normalnym [8]:

$$z = \frac{p_A - p_B}{\sqrt{2p(1-p)/N}}, \tag{18}$$

gdzie $p = (p_A + p_B)/2$.

Test dwustronny polega na wyznaczeniu prawdopodobieństwa, z jakim zmienna losowa o rozkładzie normalnym może przybierać wartości znajdujące się wewnątrz wyznaczonego przedziału[21] (najczęściej przedział tworzony jest za pomocą wartości bezwzględnej liczby):

$$P(|a| \leqslant z) = 2\psi_0(|z|) - 1, \tag{19}$$

$\psi_0(z)$ jest dystrybuantą rozkładu normalnego zmiennej z. Podstawiając założone prawdopodobieństwo:

$$\bar{p} = 0.95 = 2\psi_0(|z|) - 1$$

można obliczyć wartość graniczną z. Wielkości te są stablicowane i dla $p{=}0.95$ otrzymujemy $z{=}1.96$.

Chcąc obliczyć przedział ufności dla wybranego klasyfikatora (czyli ustalonej wartości klasyfikacji) zakładamy istnienie w otoczeniu badanego klasyfikatora C innego modelu (H) i odwracamy sytuację: zamiast badać statystyczną rozróżnialność klasyfikatorów w danym przedziale, szukamy przedziału, gdzie z określonym prawdopodobieństwem nie znajdziemy innych klasyfikatorów. Korzystając z wzorów (18) i (19) tworzymy nierówności [26]:

$$-z \leqslant \frac{p_C - p_H}{\sqrt{2p(1-p)/N}} \leqslant z,$$

których rozwiązania względem p_H wyznaczają parę punktów:

$$\frac{c + z^2/2N \pm z\sqrt{c/n - c^2/N + z^2/4N^2}}{1 + z^2/N},$$

[21] Sytuację można oczywiście odwrócić i badać prawdopodobieństwo znalezienia się zmiennej poza przedziałem.

gdzie $c = 1 - p_C$ jest dokładnością klasyfikacji modelu C. Punkty te tworzą przedział ufności, poza którym z prawdopodobieństwem 0.05 znajdziemy klasyfikatory lepszy i gorszy.

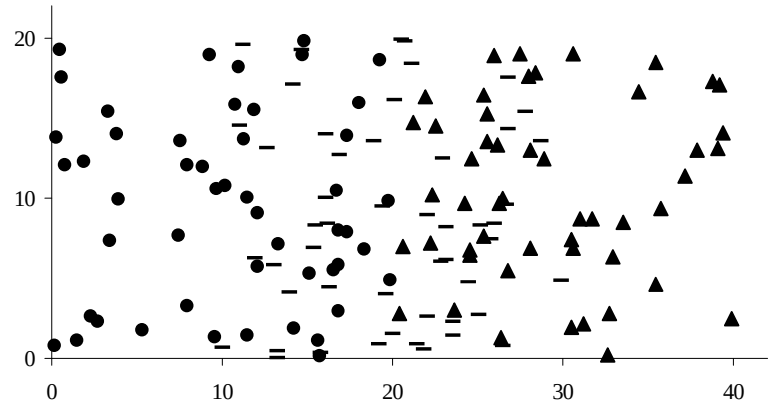
W pracy zamieszczono przedziały ufności dla końcowej klasyfikacji komitetu CUC na zbiorze testowym dla $\bar{p} = 0.95$ oraz obliczono maksymalne zaufanie p_{MAX}, dla którego wynik komitetu CUC jest lepszy od wyniku najbliższego (biorąc pod uwagę dokładność) komitetu. Wyznaczanie p_{MAX} polega na poszukiwaniu takiego najszerszego przedziału (manipulując wielkością $\bar{p}$), w którym nie będzie się zawierał wynik innego komitetu.

4.3.1 Zbiór danych sztucznych

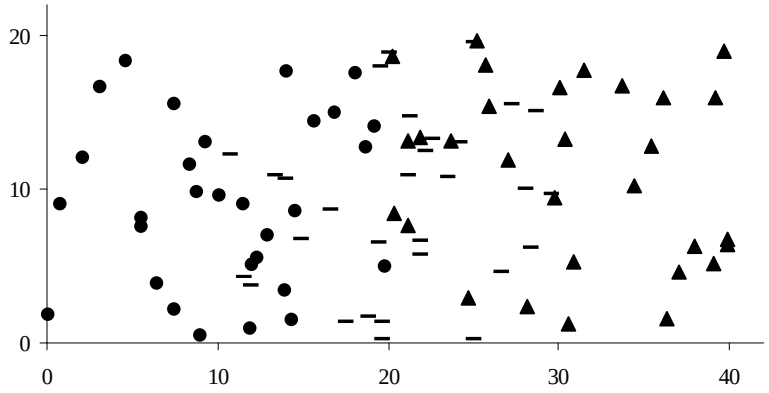
Celem przetestowania, przeanalizowania i lepszego zrozumienia pracy komitetu, stworzono dwuwymiarowe (cechy ciągłe), trzyklasowe sztuczne dane składające się z dwóch niezależnych zbiorów przedstawionych na rysunkach 35 i 36, o rozkładzie klas zawartym w poniższej tabeli:

	C_1	C_2	C_3	Razem
$\mathcal{L}$	50	50	50	**150**
$\mathcal{T}$	30	30	30	**90**

Dane wygenerowano na podstawie zbioru losowych punktów wypełniających trzy nakładające się na siebie kwadraty o boku równym 20.



Rysunek 35: Zbiór treningowy danych sztucznych ($N = 150$).



Rysunek 36: Zbiór testowy danych sztucznych ($N = 90$).

Modele wybrane do komitetu:

Nazwa	Opis
M_1	*FSM*
M_2	*SSV*
M_3	*IncNet*
M_4	kNN, $k = 5$, Euklides

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{L}$):

	M_1	M_2	M_3	M_4
C_1	90.00	94.00	94.00	74.00
C_2	56.00	68.00	50.00	62.00
C_3	90.00	86.00	46.00	82.00
śr.	78.67	82.67	63.33	72.67

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4
C_1	83.33	83.33	93.33	66.67
C_2	23.33	40.00	20.00	50.00
C_3	96.67	76.67	16.67	80.00
śr.	67.78	66.67	43.33	65.56

Najlepsza możliwa klasyfikacja dla wybranych modeli:

	C_1	C_2	C_3	śr.
$\mathcal{L}$	100.00	96.00	98.00	98.00
$\mathcal{T}$	96.67	70.00	96.67	87.78

Otrzymana macierz współczynników $\underline{\mathbf{W}}$:

0.3369	0.7546	0.0186	-0.1268
0.3650	0.6508	0.3794	-0.1267
0.3492	0.5181	0.3325	-0.0756

Wynik komitetu CUC:

Opis komitetu	$\mathcal{L}$	$\mathcal{T}$
CUC, $a=7$, $\lambda=0.95$	**97.33**	**72.22**
Głosowanie większościowe	81.00	68.34
Komitet największego zaufania	70.67	64.41

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [62.20,80.42].
$p_{MAX}=0.57$

4.3.2 Zbiór Vowel

Jest to zbiór opisujący samogłoski języka angielskiego, każda z samogłosek (reprezentująca jedną z 11-stu klas) opisywana jest za pomocą 10-ciu ciągłych cech.

Rozkład wektorów w poszczególnych klasach:

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	Razem
$\mathcal{L}$	48	48	48	48	48	48	48	48	48	48	48	**528**
$\mathcal{T}$	42	42	42	42	42	42	42	42	42	42	42	**462**

Wyniki najlepszych pojedynczych klasyfikatorów:

Nazwa	$\mathcal{L}$	$\mathcal{T}$
Square node network, 88 units	–	54.8
Gaussian node network, 528 units	–	54.6
1-NN, Euclides, raw	99.24	56.3
Radial Basis Function, 528 units	–	53.5
Gaussian node network, 88 units	–	53.5
FSM Gauss, 10CV na zbiorze $\mathcal{L}$	92.60	51.94

Modele wybrane do komitetu:

Nazwa	Opis
M_1	kNN, $k=5$, Euklides
M_2	*FSM*
M_3	kNN, $k=7$, Chebyshev
M_4	kNN, $k=7$, Euklides
M_5	kNN, $k=7$, Manhattan
M_6	kNN, $k=9$, Euklides
M_7	kNN, $k=9$, Manhattan
M_8	*IncNet*
M_9	*SSV, beam search*

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{L}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
C_1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
C_2	95.83	97.92	95.83	91.67	95.83	95.83	93.75	97.92	95.83
C_3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	97.92	95.83
C_4	97.92	100.0	97.92	100.0	97.92	100.0	97.92	97.92	95.83
C_5	93.75	97.92	72.92	95.83	95.83	83.33	91.67	95.83	93.75
C_6	81.25	93.75	66.67	72.92	68.75	66.67	66.67	66.67	91.67
C_7	85.42	97.92	72.92	79.17	81.25	70.83	72.92	97.92	93.75
C_8	97.92	100.0	95.83	97.92	97.92	100.0	95.83	100.0	97.92
C_9	91.67	100.0	91.67	91.67	89.58	81.25	81.25	95.83	100.0
C_{10}	97.92	100.0	87.50	91.67	93.75	87.50	91.67	97.92	93.75
C_{11}	100.0	100.0	100.0	97.92	97.92	91.67	89.58	100.0	97.92
śr.	94.70	98.86	89.20	92.61	92.61	88.83	89.20	95.27	96.02

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
C_1	52.38	66.67	54.76	57.14	61.90	54.76	61.90	28.57	57.14
C_2	64.29	42.86	64.29	64.29	47.62	57.14	57.14	83.33	50.00
C_3	83.33	66.67	52.38	88.10	80.95	85.71	80.95	76.19	30.95
C_4	59.52	54.76	61.90	61.90	69.05	64.29	73.81	69.05	33.33
C_5	50.00	33.33	64.29	50.00	40.48	45.24	45.24	26.19	38.10
C_6	61.90	40.48	54.76	61.90	50.00	57.14	57.14	35.71	23.81
C_7	50.00	83.33	50.00	50.00	52.38	38.10	38.10	78.57	64.29
C_8	78.57	73.81	76.19	80.95	80.95	83.33	76.19	78.57	42.86
C_9	33.33	11.90	28.57	35.71	33.33	35.71	33.33	30.95	59.52
C_{10}	33.33	47.62	30.95	26.19	26.19	23.81	26.19	16.67	57.14
C_{11}	78.57	38.10	83.33	83.33	69.05	88.10	71.43	78.57	21.43
śr.	58.66	50.87	56.49	59.96	55.63	57.58	56.49	54.76	43.51

Najlepsza możliwa klasyfikacja dla wybranych modeli:

	C_1	C_2	C_3	C_4	C_5	C_6
$\mathcal{L}$	100.00	100.00	100.00	100.00	97.92	97.92
$\mathcal{T}$	80.95	97.62	97.62	88.10	83.33	80.95

	C_7	C_8	C_9	C_{10}	C_{11}	śr.
$\mathcal{L}$	97.92	100.00	100.00	100.00	100.00	99.43
$\mathcal{T}$	97.62	92.86	73.81	85.71	95.24	88.53

Otrzymana macierz współczynników $\underline{\mathbf{W}}$:

0.540	0.115	-0.045	-0.055	-0.018	0.114	-0.166	0.292	0.233
0.389	0.398	-0.008	-0.123	-0.021	0.071	0.001	0.053	0.316
0.427	0.481	-0.048	-0.066	0.072	-0.014	-0.042	0.046	0.162
0.084	1.011	-0.093	0.189	-0.020	-0.140	-0.008	-0.049	0.039
0.142	0.554	-0.028	-0.122	0.127	0.018	-0.162	0.237	0.310
0.170	0.790	-0.014	-0.138	0.126	0.012	-0.099	-0.073	0.297
0.052	0.465	0.006	-0.042	-0.042	-0.089	0.003	0.414	0.321
0.194	0.471	-0.002	-0.197	0.120	0.052	-0.161	0.087	0.443
0.062	0.736	-0.028	-0.034	0.035	-0.038	-0.056	0.220	0.215
0.298	0.602	-0.046	0.064	-0.025	0.013	-0.145	0.165	0.117
0.088	1.006	-0.041	0.029	0.030	-0.057	-0.111	-0.091	0.188

Wynik komitetu *CUC*:

Opis komitetu	$\mathcal{L}$	$\mathcal{T}$
CUC, $a=5$, $\lambda=0.95$	**99.24**	**62.12**
Głosowanie większościowe	94.80	61.82
Komitet największego zaufania	94.51	47.94

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [57.61,66.43].
$P_{MAX} = 0.105$

4.3.3 Zbiór DNA

Zbiór ten pochodzi z projektu analizy sekwencji kodu DNA i służy do określania fragmentów kodu, w których znajdują się obszary exon/intron, obszary intron/exon i obszary niekodujące żadnych białek. Pojedynczy wektor pochodzący z oryginalnego zbioru jest ciągiem złożonym z 60-ciu symboli {a,c,t,g}. Ponieważ dane symboliczne mogą być w sposób ograniczony analizowane numerycznie, zbiór ten został poddany transformacji:

$$P(C_j|x_f) = \frac{N_j(x_f)}{N(x_f)}$$

$(j = 1, \ldots, K, \;\; f = 1, \ldots, n)$ zastępującej symbole prawdopodobieństwami. $N_j(x_f)$ jest liczbą wystąpień wartości x_f danej cechy w j-ej klasie, natomiast $N(x_f)$ liczbą wszystkich wystąpień wartości tej samej cechy. Powstały w ten sposób wektor składa się ze 180-ciu ciągłych atrybutów, liczba klas się nie zmienia i wynosi 3.

Rozkład wektorów w poszczególnych klasach:

	C_1	C_2	C_3	Razem
$\mathcal{L}$	464	485	1051	**2000**
$\mathcal{T}$	303	280	603	**1186**

Wyniki najlepszych pojedynczych klasyfikatorów:

Nazwa	$\mathcal{L}$	$\mathcal{T}$
RBF 720 nodes	98.5	95.9
kNN, $P(X\|C)$, k=6, Euclides	96.8	95.5
Dipol92	99.3	95.2
Alloc80	93.7	94.3
QuaDisc	100.0	94.1
LDA, Discrim	96.6	94.1

Modele wybrane do komitetu:

Nazwa	Opis
M_1	kNN, k=5, Euklides
M_2	kNN, k=7, Euklides
M_3	*FSM*
M_4	*IncNet*
M_5	*SSV*, *beam search*
M_6	*SSV*, *beam search*, optymalne przycinanie

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{L}$):

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	94.83	96.34	96.77	91.16	83.41	80.60
C_2	96.70	96.49	97.73	91.96	85.57	89.07
C_3	94.48	93.43	96.76	98.19	87.63	94.96
śr.	95.10	94.85	97.00	95.05	86.15	90.20

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	96.70	97.69	94.39	91.09	76.57	74.26
C_2	95.36	96.07	92.50	87.50	85.00	84.64
C_3	94.36	93.70	95.52	98.51	89.72	95.36
śr.	95.19	95.28	94.52	94.01	85.24	87.44

Najlepsza możliwa klasyfikacja dla wybranych modeli:

	$\mathcal{L}$	$\mathcal{T}$
C_1	99.14	98.68
C_2	98.76	97.86
C_3	98.38	99.00
śr.	98.65	98.65

Otrzymana macierz współczynników $\underline{\mathbf{W}}$:

-0.0428	0.0938	0.6917	0.3443	-0.1063	0.0572
0.1904	-0.2377	0.7031	0.4271	-0.0408	0.0047
0.1433	0.0394	0.5762	0.3971	-0.1183	-0.0457

Wynik komitetu *CUC*:

Opis komitetu	$\mathcal{L}$	$\mathcal{T}$
CUC, $a=9$, $\lambda=0.95$	**98.05**	**95.70**
Głosowanie większościowe	96.58	94.73
Komitet największego zaufania	99.63	92.71

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [94.39,96.71].
$P_{MAX}=0.86$

4.3.4 Zbiór Letters

Klasy zawarte w zbiorze symbolizują 26 liter alfabetu, ich etykiety mają zostać przewidziane na podstawie 16-stu cech (powstałych w wyniku przekształceń danych pochodzących z procesu optycznego rozpoznawania pisma).

Rozkład wektorów w poszczególnych klasach:

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
$\mathcal{L}$	594	567	554	598	565	565	547	538	567
$\mathcal{T}$	195	199	182	207	203	210	226	196	188

	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}
$\mathcal{L}$	575	572	560	607	605	574	597	594	568
$\mathcal{T}$	172	167	201	185	178	179	206	189	190

	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	Razem
$\mathcal{L}$	563	599	606	581	576	583	602	543	**15000**
$\mathcal{T}$	185	197	207	183	176	204	184	191	**5000**

Wyniki najlepszych pojedynczych klasyfikatorów:

Nazwa	$\mathcal{L}$	$\mathcal{T}$
ALLOC80	93.5	93.6
kNN	100.00	93.2
LVQ	94.3	92.1
Quadisc	89.9	88.7
CN2	97.9	88.5
Baytree	98.5	87.6

Modele wybrane do komitetu:

Nazwa	Opis
M_1	kNN, $k=5$, Euklides
M_2	kNN, $k=9$, Euklides
M_3	*FSM*
M_4	kNN, $k=7$, Euklides
M_5	*SSV*, optymalne przycianie
M_6	kNN, $k=5$, Manhattan
M_7	kNN, $k=11$, Euklides

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{L}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
C_1	98.99	97.81	98.65	98.82	92.93	98.99	97.31
C_2	93.65	93.12	97.18	93.65	75.31	94.53	92.95
C_3	95.31	95.13	98.56	95.49	83.21	94.40	94.95
C_4	94.15	93.48	97.66	93.65	80.27	95.65	93.14
C_5	94.51	94.51	94.87	94.51	76.46	94.69	94.34
C_6	91.68	89.73	96.11	91.68	82.48	92.21	89.20
C_7	93.24	91.77	96.53	92.14	77.15	93.24	90.86
C_8	84.39	81.23	92.94	82.90	66.17	86.25	81.41
C_9	95.94	95.41	97.18	95.59	81.83	95.24	95.59
C_{10}	93.57	92.52	97.22	93.22	84.35	93.74	92.17
C_{11}	90.73	88.81	97.20	90.03	76.05	91.43	88.29
C_{12}	96.61	95.89	97.32	96.43	80.71	96.61	95.89
C_{13}	95.88	94.07	98.85	94.40	83.86	95.39	93.90
C_{14}	93.72	92.07	98.35	93.22	83.31	92.23	91.57
C_{15}	95.64	94.43	96.52	94.95	80.14	96.69	94.60
C_{16}	92.13	91.46	96.98	92.13	85.26	93.47	90.79
C_{17}	95.29	94.78	98.15	94.95	78.45	95.45	93.60
C_{18}	94.01	94.01	97.36	94.37	75.18	92.78	92.25
C_{19}	96.27	95.91	99.29	96.45	72.29	96.45	95.20
C_{20}	95.83	95.83	97.83	96.16	85.48	96.33	95.66
C_{21}	97.52	97.03	98.02	97.03	86.96	97.85	97.52
C_{22}	96.21	94.49	96.39	95.18	90.36	96.21	94.84
C_{23}	97.22	97.40	99.65	97.57	90.63	97.92	96.70
C_{24}	95.71	94.68	97.94	95.03	80.10	96.57	94.34
C_{25}	98.01	97.34	97.67	97.67	78.57	97.84	96.51
C_{26}	98.34	97.97	99.08	98.34	80.66	97.42	96.87
śr.	94.83	93.92	97.46	94.48	81.19	95.02	93.52

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
C_1	97.95	97.44	97.44	97.95	87.69	97.44	97.95
C_2	94.97	94.47	91.46	95.48	71.86	95.98	94.97
C_3	95.05	95.05	93.41	95.05	80.22	95.05	94.51
C_4	99.52	98.55	95.17	99.03	74.40	97.10	98.07
C_5	92.12	92.12	86.70	91.63	68.47	93.60	92.12
C_6	93.81	92.86	86.67	93.33	79.05	90.00	92.86
C_7	93.81	90.71	90.27	92.48	71.68	92.48	91.15
C_8	89.80	87.24	85.71	87.76	71.43	89.29	86.22
C_9	93.09	94.68	93.09	93.09	75.53	93.62	94.68
C_{10}	93.02	93.60	93.02	93.60	84.30	93.02	93.02
C_{11}	88.02	86.83	88.62	88.02	68.86	91.02	86.23
C_{12}	97.51	97.51	94.03	97.51	79.60	97.51	96.02
C_{13}	97.84	96.22	97.84	97.30	80.00	96.22	95.68
C_{14}	96.63	95.51	98.31	96.07	83.15	96.07	95.51
C_{15}	96.65	94.97	87.15	96.09	73.74	97.77	94.97
C_{16}	95.15	93.20	96.12	95.15	83.01	95.63	92.72
C_{17}	94.18	94.71	95.24	94.18	71.96	94.18	94.18
C_{18}	91.58	90.53	86.84	90.53	72.63	93.16	91.58
C_{19}	97.30	95.68	94.05	96.22	70.27	95.68	95.14
C_{20}	98.98	96.95	95.43	97.97	78.68	98.98	98.48
C_{21}	98.55	98.07	95.17	98.07	85.99	97.58	98.07
C_{22}	98.36	98.36	93.44	98.36	86.34	97.27	98.91
C_{23}	97.73	97.16	97.73	96.59	85.23	97.16	97.16
C_{24}	93.14	91.67	95.59	92.16	75.49	92.65	91.67
C_{25}	99.46	99.46	94.57	99.46	80.43	97.83	99.46
C_{26}	96.86	97.91	94.24	97.38	74.35	97.91	96.86
śr.	95.44	94.66	92.94	95.02	77.42	95.14	94.54

Najlepsza możliwa klasyfikacja dla wybranych modeli:

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
$\mathcal{L}$	100.0	99.29	99.10	99.50	99.12	99.47	98.54	96.65	98.94
$\mathcal{T}$	100.0	99.50	97.25	100.0	99.01	97.14	96.46	95.41	95.21

	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}
$\mathcal{L}$	98.78	98.95	98.39	99.51	99.34	99.30	98.83	99.33	99.12
$\mathcal{T}$	97.67	97.60	98.51	100.0	100.0	99.44	99.03	97.88	98.42

	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	śr.
$\mathcal{L}$	100.0	99.33	99.67	98.97	99.83	99.66	99.83	100.0	99.22
$\mathcal{T}$	100.0	100.0	99.52	99.45	99.43	100.0	100.0	99.48	98.70

Otrzymana macierz współczynników $\underline{\mathbf{W}}$:

0.9605	-0.3310	0.2114	-0.0025	0.0349	0.4903	-0.2931
0.5848	0.0329	0.7480	-0.0181	0.0187	0.2016	-0.2804
0.7044	0.0257	0.2551	0.1765	-0.0040	0.2433	-0.3313
0.3746	0.0068	0.5363	0.2175	0.0224	0.3851	-0.3380
0.5858	0.1344	0.6642	-0.0586	-0.0057	0.3145	-0.3761
0.6269	-0.1290	0.3933	0.2414	0.1855	0.3252	-0.4865
0.5852	-0.0478	0.3856	0.1445	0.0393	0.3596	-0.2987
0.4877	0.1707	0.9597	-0.2324	-0.0139	0.3020	-0.3022
0.4592	0.6762	0.5971	0.0781	-0.0117	0.0666	-0.6842
0.7083	0.2420	0.2309	-0.0293	0.0238	0.3368	-0.4695
0.4674	-0.0888	0.8164	-0.0467	0.0562	0.4130	-0.3372
1.0235	0.0274	0.1709	-0.0990	-0.0539	0.3217	-0.3636
0.9224	0.1294	0.3830	-0.2198	0.0092	0.2499	-0.4155
0.8331	0.0226	0.3901	0.0824	-0.0246	0.1802	-0.3781
0.6511	-0.0444	0.6851	-0.0422	0.0143	0.3115	-0.2954
0.4254	-0.1275	0.2924	0.2211	0.1248	0.5268	-0.3792
0.5757	-0.2347	0.3162	0.1817	0.0603	0.3905	-0.1876
0.4824	-0.1451	0.6232	0.2869	0.0191	0.3208	-0.3698
0.7156	-0.1117	0.2195	0.2044	0.0213	0.3718	-0.3302
0.5997	0.1662	0.3213	0.0279	0.1118	0.2819	-0.4044
0.8531	-0.0114	0.2041	-0.1712	0.0321	0.4239	-0.2653
0.7754	0.0409	0.2041	-0.3016	0.1047	0.3983	-0.1496
0.7302	-0.0148	0.1998	-0.0865	0.0096	0.5919	-0.3754
0.5464	-0.0415	0.4320	-0.0375	0.1009	0.4445	-0.2391
0.6308	0.0513	0.2701	-0.0648	0.0997	0.3540	-0.2335
0.8371	0.1049	0.3774	-0.0623	0.0019	0.1200	-0.3083

Wynik komitetu *CUC*:

Opis komitetu	$\mathcal{L}$	$\mathcal{T}$
CUC, $a=5$, $\lambda=0.95$	**98.45**	**96.52**
Głosowanie większościowe	95.82	95.41
Komitet największego zaufania	93.92	92.62

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [95.98,96.99].
$P_{MAX}=0.9998$

4.3.5 Zbiór Satelite Image

Jest to zbiór danych numerycznych powstałych w wyniku analizy zdjęć satelitarnych Ziemi, w którym w klasyfikacji danego przypadku (będącego fragmentem większej części zdjęcia) mają pomóc cechy wektorów jego najbliższego otoczenia (pomysł oparty na założeniu, iż sąsiednie wektory mają takie same klasy). W zbiorze istnieje 6 klas[22], wektory składają się z 36-ciu cech.

Rozkład wektorów w poszczególnych klasach:

	C_1	C_2	C_3	C_4	C_5	C_6	Razem
$\mathcal{L}$	1072	479	961	415	470	1038	**4435**
$\mathcal{T}$	461	224	397	211	237	470	**2000**

Wyniki najlepszych pojedynczych klasyfikatorów:

Nazwa	$\mathcal{L}$	$\mathcal{T}$
kNN	91.1	90.6
kNN, k=2,3, Euklides	–	90.3
LVQ	95.2	89.5
DIPOL92	94.9	88.9
RBF	88.9	87.9
ALLOC80	96.4	86.8

Modele wybrane do komitetu:

Nazwa	Opis
M_1	*FSM*
M_2	*SSV, beam search*
M_3	kNN, $k=9$, Euklides
M_4	kNN, $k=9$, Manhattan
M_5	kNN, $k=5$, Euklides
M_6	kNN, $k=7$, Euklides
M_7	kNN, $k=5$, Manhattan

[22]Pierwotnie w zbiorze wydzielono 7 klas, jednak później usunięto klasę 6-tą. Tu, dla ciągłości numeracji, klasę 7 oznaczono jako 6.

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{L}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
C_1	99.16	94.96	97.57	98.32	97.67	97.57	98.04
C_2	99.16	86.22	96.87	97.08	97.49	96.87	97.49
C_3	99.27	95.11	95.21	96.15	93.55	94.28	94.17
C_4	58.80	40.96	62.89	64.82	68.19	65.54	66.99
C_5	90.21	64.47	84.26	83.83	86.81	86.38	86.60
C_6	96.72	86.13	89.40	88.34	88.82	89.21	88.92
śr.	93.89	83.70	90.42	90.71	90.78	90.64	90.89

Klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
C_1	98.92	94.79	99.35	99.57	98.92	99.13	99.13
C_2	97.77	90.18	97.32	96.43	96.88	96.88	96.43
C_3	97.23	95.47	93.45	94.46	93.20	93.20	92.70
C_4	42.65	40.28	64.93	63.03	69.19	65.88	66.35
C_5	81.86	59.92	83.54	84.39	87.34	87.34	86.92
C_6	92.55	82.98	85.96	86.81	87.45	86.17	88.51
śr.	89.00	81.75	89.30	89.55	90.35	89.75	90.15

Najlepsza możliwa klasyfikacja dla wybranych modeli:

	C_1	C_2	C_3	C_4	C_5	C_6	śr.
$\mathcal{L}$	99.72	99.37	99.58	78.31	95.53	97.88	96.78
$\mathcal{T}$	99.78	99.11	98.99	76.78	94.09	95.96	95.55

Otrzymana macierz współczynników $\underline{\mathbf{W}}$:

0.3347	-0.0545	-0.2501	0.1882	1.1212	-0.3792	0.0704
0.9560	-0.0995	-0.1758	0.1065	0.2368	-0.1874	0.1651
1.0226	-0.1499	-0.1346	0.0332	0.2338	-0.1102	0.1664
1.4888	-0.1362	-0.3425	-0.2504	0.2244	0.0391	0.3930
0.9736	-0.2085	-0.2977	0.0695	0.1989	0.2665	0.1686
0.9289	-0.1403	-0.2932	-0.3121	0.1661	0.2969	0.4080

Wynik komitetu *CUC*:

Opis komitetu	$\mathcal{L}$	$\mathcal{T}$
CUC, $a=13$, $\lambda=0.95$	**94.99**	**91.10**
Głosowanie większościowe	93.27	89.6
Komitet największego zaufania	93.05	88.95

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [89.77,92.27].
$p_{MAX}=0.972$

4.3.6 Zbiór Telugu Vowel

Dane składają się wektorów podzielonych na 6 klas (symbolizujących głoski Indian Telugu), każdy wektor posiada 3 składowe (wyodrębnione na podstawie częstotliwości głosek).

Obliczenia dla tego zbioru zostały przeprowadzone wcześniej i zamieszczone w pracy [13]. Ponieważ zbiór ten nie posiada części testowej, obliczenia przeprowadzane były w trybie pięciokrotnej $2 \times CV$, następnie uśredniane. Z tego powodu niemożliwe było zamieszczenie wszystkich wyników.

Rozkład wektorów w poszczególnych klasach:

C_1	C_2	C_3	C_4	C_5	C_6	Razem
72	89	172	151	207	180	**871**

Wyniki najlepszych pojedynczych klasyfikatorów:

Nazwa	$\mathcal{L}$	$\mathcal{T}$
kNN	91.1	90.6
kNN, $k=2,3$, Euklides	–	90.3
LVQ	95.2	89.5
DIPOL92	94.9	88.9
RBF	88.9	87.9
ALLOC80	96.4	86.8

Modele wybrane do komitetu:

Nazwa	Opis
M_1	kNN, $k=10$, Euklides
M_2	kNN, $k=13$, Manhattan
M_3	kNN, $k=5$, Euklides
M_4	kNN, $k=5$, Manhattan

Średnia klasyfikacja za pomocą wybranych modeli w poszczególnych klasach ($\mathcal{T}$):

	M_1	M_2	M_3	M_4
C_1	50.0	45.8	65.3	62.5
C_2	88.8	91.0	87.6	89.9
C_3	84.3	84.3	84.9	84.7
C_4	85.4	84.8	90.1	88.1
C_5	91.3	88.4	90.3	90.1
C_6	90.6	92.8	90.1	90.4
śr.	85.1	84.6	86.1	86.0

Najlepsza możliwa klasyfikacja ($\mathcal{T}$): 90.1

Wynik komitetu *CUC*:

Opis komitetu	$\mathcal{T}$
CUC, $a=5$, $\lambda=0.95$	**88.2**
Komitet największego zaufania	87.0
Głosowanie większościowe	85.9

Przedział ufności dla $\bar{p}=0.95$ ($\mathcal{T}$): [85.89,90.18].
$P_{MAX}=0.708$

Zakończenie

W pracy zaprezentowano nowatorski sposób tworzenia komitetu, który odzwierciedla współczesne tendencje dominujące w dziedzinie zbiorowych klasyfikatorów. Przedstawiono jego założenia teoretyczne oraz rezultaty testów na popularnych zbiorach danych. Uzyskane wyniki potwierdzają przewagę komitetu *CUC* nad prostymi komitetami, przeprowadzenie większych porównań było jednak niemożliwe.

Stworzony projekt nie jest zapewne narzędziem uniwersalnym, ale także ukończonym. Wiele etapów pracy wykonywanych ręcznie (głównie wybór członków komitetu) wymaga dalszego rozwoju i automatyzacji. Problemy, które pojawiły się w trakcie pracy (niestabilności numeryczne wynikające z osobliwych układów równań, z których wyznaczano macierze współczynników $\underline{\mathbf{W}}$), wymagają nowych pomysłów i rozwiązań.

Także działanie komitetu nie zostało do końca przeanalizowane. Dalsze badania powinny dotyczyć wpływu liczby członków komitetu oraz parametrów i kształtów funkcji niekompetencji na końcowy wynik klasyfikacyjny.

Projekt ten może być udoskonalany na wiele sposobów z racji dwuczęściowej budowy: pierwsza odpowiada za obliczanie współczynników występujących w kombinacji liniowej, druga część zajmuje się wyznaczaniem modeli mających brać udział w końcowej decyzji (poprzez wprowadzenie funkcji niekompetencji). Każda z części może być niezależnie udoskonalana i współpracować z drugą, obie doczekały się już modyfikacji i będą przedmiotem dalszych eksperymentów. Równie interesujące wydaje się zastosowanie metod opartych na podobieństwie do określania obszarów niekompetencji – w najprostszej wersji dla każdego wektora testowego wyszukujemy najbliższy poprawnie klasyfikowany wektor treningowy, sprawdzamy który model dokonał poprawnej analizy i używamy go do klasyfikacji wektora testowego. Naturalnym rozszerzeniem tej metody jest wyszukiwanie kilku najbliższych wektorów i stosowanie wśród wybranych modeli głosowania większościowego. To tylko nieliczne z wielu pomysłów czekających na realizację. Badania trwają ...

Dodatek A

Rozwiązywanie nadokreślonych układów równań liniowych

Nadokreślonym układem równań liniowych $\underline{\mathbf{A}}\mathbf{x}=\mathbf{b}$ nazywamy układ postaci:

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}\right\} \tag{1}$$

przedstawiony na rysunku 37, w którym ilość równań wiążąca niewiadome $\mathbf{x}$ jest większa od ilości niewiadomych. Taki układ nie posiada jednego ścisłego rozwiązania, istnieje wiele sposobów wyznaczenia „najlepszego" wektora $\mathbf{x}$. Jednym z nich jest tzw. rozwiązanie średniokwadratowe.

Według [7], rozwiązanie średniokwadratowe układu nadokreślonego polega na wyznaczeniu wektora $\mathbf{x}$ minimalizującego długość euklidesową wektora residuum $\mathbf{r}$:

$$\|\mathbf{r}\| = \sqrt{\mathbf{r}^T\mathbf{r}}\,, \quad \text{gdzie} \quad \mathbf{r}=\mathbf{b}-\underline{\mathbf{A}}\mathbf{x}.$$

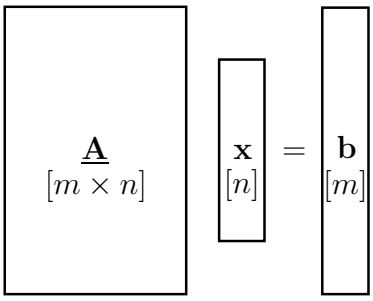
Cel ten osiąga się poprzez sprowadzenie układu do tzw. postaci normalnej, czyli do układu o rozmiarach $n \times n$.

Mnożąc lewostronnie układ (1) przez macierz transponowaną $\underline{\mathbf{A}}^T$:

$$\underline{\mathbf{A}}^T\underline{\mathbf{A}}\mathbf{x}=\underline{\mathbf{A}}^T\mathbf{b}, \tag{2}$$

otrzymujemy, wprowadzając nowe zmienne:

$$\underline{\mathbf{A}}'\mathbf{x} = \mathbf{b}', \quad \text{gdzie} \quad \underline{\mathbf{A}}'=\underline{\mathbf{A}}^T\underline{\mathbf{A}}, \;\; \mathbf{b}'=\underline{\mathbf{A}}^T\mathbf{b}. \tag{3}$$



Rysunek 37: Ogólna postać liniowego układu nieokreślonego o wymiarach $m \times n$ $(m > n)$.

Jest to układ równań liniowych z kwadratową macierzą $\underline{\mathbf{A}}$, który można rozwiązać jedną z bezpośrednich metod znanych z analizy numerycznej, np. metodą Doolittle'a z częściowym wyborem elementu głównego. Metoda ta polega na rozkładzie macierzy $\underline{\mathbf{A}}'$ na iloczyn macierzy $\underline{\mathbf{LU}}$ (macierzy trójkątnej dolnej i trójkątnej górnej) wykorzystując tzw. schematy zwarte eliminacji Gaussa. Elementy macierzy $\underline{\mathbf{L}}$ oraz $\underline{\mathbf{U}}$ otrzymujemy z wzorów [7]:

$$\begin{aligned} l_{ik} &= \frac{1}{u_{kk}}\left\{a'_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk}\right\} \qquad && i = k+1, k+2, \ldots, n \\ u_{kj} &= a'_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj} && j = k, k+1, \ldots, n \end{aligned} \quad , \tag{4}$$

gdzie k jest krokiem obliczeń, $k = 1, 2, \ldots, n$. Opierając się na założeniach:

$$\begin{cases} \underline{\mathbf{A}}'\mathbf{x} = \mathbf{b}' \\ \underline{\mathbf{A}}' = \underline{\mathbf{LU}} \end{cases} \quad \Rightarrow \quad \begin{cases} \underline{\mathbf{L}}\mathbf{y} = \mathbf{b}' \\ \underline{\mathbf{U}}\mathbf{x} = \mathbf{y} \end{cases} ,$$

do rozwiązania pozostają dwa układy równań z macierzami trójkątnymi. Mając wyznaczone macierze $\underline{\mathbf{L}}$ oraz $\underline{\mathbf{U}}$ układy te rozwiązujemy stosując (w kolejności) podstawianie w przód i wstecz:

$$\begin{aligned} y_i &= \frac{1}{l_{ii}}\left\{b'_i - \sum_{k=1}^{i-1} l_{ik}y_k\right\} \qquad && i = 1, 2, \ldots, n \\ x_i &= \frac{1}{u_{ii}}\left\{y_i - \sum_{k=i+1}^{n} u_{ik}x_k\right\} && i = n, n-1, \ldots, 1 \end{aligned} \tag{5}$$

Otrzymany z (5) wektor $\mathbf{x}$ jest średniokwadratowym rozwiązaniem układu równań (1).

Literatura

[1] Adamczak R.: *Zastosowanie sieci neuronowych do klasyfikacji danych doswiadczalnych.* Praca doktorska, Uniwersytet Mikołaja Kopernika, Katedra Metod Komputerowych, Toruń, 2001.

[2] Brandt S.: *Analiza danych.* Wydawnictwo Naukowe PWN, Warszawa, 1998.

[3] Breiman L.: *Arcing Classifiers.* Technical Report 460, Department of Statistics, University of California, Berkeley, CA, 1996.

[4] Breiman L.: *Bagging predictors.* Technical Report 421, Department of Statistics, University of California, Berkeley, CA, 1994.

[5] Burden R. L., Faires J. D.: *Numerical Analysis. Third edition.* Prindle, Weber & Schmidt, Boston, 1985.

[6] Cichosz P.: *Systemy uczące się.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[7] Dahlquist G., Björck A.: *Metody numeryczne.* Wydawnictwo Naukowe PWN, Warszawa, 1983.

[8] Dietterich T. G.: *Approximate statistical tests for comparing supervised classification learning algorithms.* Neural Computation, 10 (7), 1998, pp. 1895-1924.

[9] Dietterich T. G.: *Ensemble Methods in Machine Learning.* First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, New York, Springer Verlag, 2000, pp. 1-15.

[10] Duch W., Grąbczewski K.: *A general purpose separability criterion for classification systems.* 4th Conference on Neural Networks and Their Applications, Zakopane, 1999, pp. 203-208.

[11] Duch W., Grudziński K.: *Sieci Neuronowe i Uczenie Maszynowe: próba integracji.* Biocybernetyka 2000, Tom 6: Sieci neuronowe (red. W. Duch, J. Korbicz, L. Rutkowski i R. Tadeusiewicz), rozdz. III.21, pp. 663-690.

[12] Duch W., Itert Ł.: *A posteriori corrections to classification methods.* International Conference on Neural Networks and Soft Computing (ICNNSC), Advances in Soft Computing, Physica Verlag (Springer) 2002. (w druku)

[13] Duch W., Itert Ł.: *Competent undemocratic committees.* International Conference on Neural Networks and Soft Computing (ICNNSC), Advances in Soft Computing, Physica Verlag (Springer) 2002. (w druku)

[14] Duda O., Hart P. E.: *Pattern Classification. 2nd edition.* John Wiley & Sons, Inc., 2001.

[15] *Electronic Textbook* © Copyright StatSoft, Inc., 1984-2002, www.statsoftinc.com/textbook/

[16] Freund Y., Schapire R. E.: *A Short Introduction to Boosting.* Journal of Japanese Society for Artificial Intelligence, 14(5), September, 1999, pp. 771-780.

[17] Freund Y., Schapire R. E.: *Experiments with a New Boosting Algorithm.* Machine Learning: Proceedings of the Thirteenth International Conference, 1996.

[18] Gama P. J. M.: *Combining Classification Alorithms.* Ph.D. thesis, Depertamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto, 1999.

[19] GhostMiner ®1.0 help, FQS Poland, www.fqspl.com.pl

[20] Giacinto G.: *Design of multiple classifier systems.* Ph.D. thesis, Ingegneria Dell'Informazione, Elettromagnetismo Applicato e Telecomunicazioni, Università Degli Studi di Salerno, 1998.

[21] Grąbczewski K., Duch W.: *The separability of split value criterion.* 5th Conference on Neural Networks and Soft Computing, Zakopane, June 2000, pp. 201-208.

[22] Hansen J. V.: *Combining Predictors. Meta Machine Learning Methods and Bias/Variance & Ambiguity Decompositions.* Ph.D. thesis, Department of Computer Science, University of Aarhus, Denmark, 2000.

[23] *Introduction to Data Mining and Knowledge Discovery, Third Edition.* © 1999 by Two Crows Corporation, www.twocrows.com

[24] Jankowski N.: *Ontogeniczne sieci neuronowe w zastosowaniu do klasyfikacji danych medycznych.* Praca doktorska, Uniwersytet Mikołaja Kopernika, Katedra Metod Komputerowych, Toruń, 1999.

[25] Kecman V.: *Learning and Soft Computing. Support Vector Machines, Neural Networks, and Fuzzy Logic Models.* The MIT Press, 2001.

[26] Kohavi R.: *A study of cross-validation and bootstrap for accuracy estimation and model selection.* Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, 1995, pp. 1137-1143.

[27] Korbicz J., Obuchowicz A., Uciński D.: *Sztuczne sieci neuronowe. Podstawy i Zastosowania.* Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1994.

[28] Liu Y., Yao X.: *Ensemble learning via negative correlation.* Neural Networks, 12(10), 1999, pp. 1399-1404.

[29] Merz C. J.: *Combining classifiers using correspondence analysis.* Advances in Neural Information Processing, Vol. 10, The MIT Press, 1998.

[30] Michie D., Spiegelhalter D. J., Taylor C. C.: *Machine learning, neural and statistical classification.* Elis Horwood, London, 1994.

[31] Neural Network FAQ, Copyright 1997-2001 by Warren S. Sarle, Cary, NC, USA, ftp://ftp.sas.com/pub/neural/FAQ.html

[32] Ortega J., Koppel M., Argamon A.: *Arbitraiting Among Competing Classifiers Using Learned Referees.* Knowledge and Information Systems, 3, 2001, pp. 470-490.

[33] Osowski S.: *Sieci neuronowe w ujęciu algorytmicznym.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1996.

[34] Schölkopf B., Smola A. J.: *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press Cambridge, Massachusetts, London, England, 2002.

[35] Skalak D. B.: *Prototype Selection For Composite Nearest Neighbor Classifiers.* Ph.D. thesis, University of Massachusetts in Amherst, Department of Computer Science, 1997.

[36] Tadeusiewicz R.:, *Sieci neuronowe.* Akademicka Oficyna Wydawnicza, Warszawa, 1993.

[37] Wolpert D.: *Stacked Generalization.* Neural Networks, 5, 1992, pp. 241–259.