KLASYFIKATOR HIERARCHICZNY

Igor T. Podolak

KLASYFIKATOR HIERARCHICZNY

z nakładającymi się grupami klas

Wydawnictwo Uniwersytetu Jagiellońskiego

Praca naukowa finansowana przez Ministerstwo Nauki i Szkolnictwa Wyższego; projekt badawczy nr 6548/B/T02/2011/40

RECENZENCI
prof. dr hab. inż. Leszek Rutkowski, czł. koresp. PAN
prof. dr hab. Mariusz Flasiński

PROJEKT OKŁADKI
Agnieszka Winciorek

ISBN 978-83-233-3444-6

WYDAWNICTWO
UNIWERSYTETU
JAGIELLOŃSKIEGO
www.wuj.pl

Wydawnictwo Uniwersytetu Jagiellońskiego
Redakcja: ul. Michałowskiego 9/2, 31-126 Kraków
tel. 12-631-18-80, tel./fax 12-631-18-83
Dystrybucja: tel. 12-631-01-97, tel./fax 12-631-01-98
tel. kom. 506-006-674, e-mail: sprzedaz@wuj.pl
Konto: PEKAO SA, nr 80 1240 4722 1111 0000 4856 3325

Spis treści

Spis rysunków

Spis tabel

Wprowadzenie

Praca przedstawia koncepcję hierarchicznego klasyfikatora HCOC (ang. *Hierarchical Classifier with Overlapping Clusters*) zbudowanego przy użyciu nowego paradygmatu automatycznego podziału na podproblemy. Wykorzystuje on przybliżoną klasyfikację przez już utworzony klasyfikator bazowy w węźle, implementowany przez prosty klasyfikator, dla podziału zadania na podproblemy. Praca przedstawia założenia teoretyczne proponowanego modelu, jego definicję, opis podstawowych zagadnień związanych z budową. W kolejnych rozdziałach zawarte są propozycje zastosowanych algorytmów wraz z uzasadnieniem teoretycznym i doświadczeniami wykazującymi ich przydatność i poprawność. Przedstawione są teoretyczne uzasadnienia poprawności modelu pokazujące jego zbieżność i przewagę nad modelami monolitycznymi.

Część z zawartego materiału ukazała się wcześniej w czasopismach informatycznych o zasięgu międzynarodowym, część prezentowana była na konferencjach i opublikowana w związanych z nimi wydawnictwach.

Praca jest częściowo finansowana przez grant Narodowego Centrum Nauki, nr NCN 6458/T02/2011/40.

Wstęp

Jednym z podstawowych problemów w nauczaniu maszynowym (ang. *Machine Learning*, ML) jest problem *klasyfikacji*. Zadanie polega na przypisaniu podanego za pomocą wektora *atrybutów* obiektu do jednej ze zdefiniowanych wcześniej *klas*. Wektor atrybutów ma zwykle skończoną, ustaloną z góry, stałą długość; liczba klas jest także skończona i ustalona.

W ramach ML zaproponowanych zostało wiele algorytmów, które rozwiązują ten problem przez *nauczanie nadzorowane*, polegające na przedstawieniu statystycznemu *algorytmowi uczącemu* wielu przykładów z przypisaną prawidłową klasą. Nauczanie tworzy *model*, który, z pewną dokładnością, potrafi rozpoznawać przykłady przedstawione wcześniej w trakcie nauczania oraz rozpoznawać część z tych, które nie zostały wcześniej pokazane. Oczywiście możliwe jest skonstruowanie takiego algorytmu, który (chociażby przez zapamiętanie wszystkich przykładów w tabeli) będzie poprawnie rozpoznawać wszystkie przykłady przedstawione w procesie nauczania. Pociąga to jednak za sobą słabsze rozpoznawanie przykładów nieprzedstawionych, czyli słabszą generalizację.

Celem ML jest znalezienie algorytmu (albo rodziny algorytmów), który będzie osiągał wysoki stopień generalizacji (tzn. będą prawidłowo rozpoznawać przykłady, które nie były uwzględnione w zbiorze uczącym). Istotne jest, aby znaleziony algorytm był jednocześnie stosunkowo prosty w użyciu, nie wymagał dodatkowej wiedzy eksperckiej, długiego czasu prób przy wyszukiwaniu suboptymalnego modelu, był maksymalnie prosty itd. Istnieje wiele podejść, takich jak systemy ekspertowe, klasyfikatory Bayesowskie, różne zastosowania logiki rozmytej, drzewa decyzyjne, algorytmy typu EM, sieci neuronowe i inne. Z ich pomocą możliwe jest budowanie podstawowych monolitycznych modeli, tzn. opartych na jednym podejściu, w odróżnieniu od modeli *hybrydowych* łączących kilka różnych podejść do utworzenia jednego modelu.

Jednym z podstawowych problemów jest to, że znalezienie optymalnego, a raczej najlepszego możliwego do znalezienia, modelu jest zwykle bardzo czasochłonne, wymaga wiedzy specjalistycznej, zależy od reprezentatywności dostępnego zbioru uczącego. Często też, aby osiągnąć wystarczająco dużą skuteczność, szukany model wymaga bardzo wielu parametrów, co wiąże się ze słabym poziomem generalizacji.

Alternatywnym podejściem jest tworzenie rozwiązań złożonych z wielu prostych modeli o niskiej skuteczności, jednak łatwych do zbudowania, których końcowe klasyfikacje są łączone. Może to polegać na podziale zbioru uczącego na szereg podprzestrzeni tworzących podproblemy. Takimi podejściami są, między innymi, *bagging*, *mixture of experts*, *uśrednianie*, *stacking*, *bumping* i inne (Hastie *et al.*, 2001; Bishop, 2006). Ważną ich cechą jest próba podejścia do częściowego rozwiązania problemu *bias–variance*.

Indywidualne modele mogą być budowane niezależnie (poprzez podział przestrzeni wejściowej) albo sekwencyjnie. Sekwencyjne podejście jest szczególnie wydajne, jeśli $k+1$ model uwzględnia wyniki działania i skuteczność poprzedniego, k-tego modelu, a stąd, pośrednio, wszystkich innych wcześniej zbudowanych. Budowany model skupia się na rozwiązaniu dla tych elementów przestrzeni problemu, które wcześniej okazały się trudne do rozwiązania. Takie podejście leży u podstaw *boostingu*, w którym budowana jest sekwencja *słabych*, bardzo prostych, modeli. Ich odpowiednie złożenie okazuje się już modelem silnym. Schapire wykazał równoważność takiego słabego nauczania z nauczaniem silnym (Schapire, 1990). Z modelu Schapire'a pochodzi, między innymi, model AdaBoost.

Modele typu AdaBoost (od ang. *Adaptive Boosting*) dzielą wejściową przestrzeń atrybutów. Robią to w sposób *miękki* przez ustanowienie rozkładu prawdopodobieństwa losowania przykładów w trakcie nauczania. Dzięki temu niektóre z modeli składowych uwzględniają w większym stopniu pewne części przestrzeni wejściowej (przestrzeni problemu), inne skupiają się na częściach, dla których te pierwsze dają słabsze wyniki. Oznacza to *podział przestrzeni wejściowej* problemu. Takie podejście pozwala na sukcesywne poprawianie rozwiązania (końcowej *hipotezy*) poprzez dodawanie nowych, prostych, klasyfikatorów i odpowiednie uśrednianie wyniku.

W tej pracy proponowany jest model, w którym dzielona jest nie przestrzeń wejściowa, lecz przestrzeń wyjściowa klas. Podział polega na wyróżnieniu grup klas, które w naturalny sposób tworzą podproblemy, a następnie rozwiązywanie tych podproblemów z osobna. Podejście to bierze swój początek z obserwacji, że dla zbioru uczącego, w którym przykłady różnych klas nie są zrównoważone, proste (słabe) klasyfikatory wykazują naturalną skłonność do grupowania wyników wokół klas przeważających w zbiorze (Podolak, 2008; Podolak, Roman, 2012). Wynika to z faktu, że niektóre klasy są bardziej do siebie podobne, a stąd modele generują podobne wzorce aktywacji. Stąd rozróżnienie, a więc i prawidłowa klasyfikacja, przykładów z klas podobnych do siebie będzie trudniejsza niż klasyfikacja przykładów z klas różniących się.

Poza klasyfikacją, możliwe jest także zastosowanie podejścia HCOC w problemach aproksymacji, co zostało pokazane w pracy (Brodowski, Podolak, 2011). W tej pracy skupimy się jednak nad samym aspektem klasyfikacji.

Klasyfikator hierarchiczny HCOC

Celem zaproponowanego modelu Hierarchicznego Klasyfikatora z Nakładającymi się Grupami Klas (ang. *Hierarchical Classifier with Overlapping Clusters*, HCOC) jest właśnie te, wspomniane wyżej, zależności wykorzystać. HCOC ma strukturę drzewa słabych klasyfikatorów, z których każdy stara się rozwiązać problem na swoim poziomie, i albo podaje końcowy wynik, albo też wskazuje na możliwość podziału aktualnego problemu na podproblemy. Każdy podproblem składa się z podzbioru klas problemu nadrzędnego, przy czym różne podproblemy jednego problemu nie muszą być rozłączne. Jest to cecha wyróżniająca zaproponowane tu podejście HCOC.

Klasyfikowany wektor atrybutów jest przedstawiany klasyfikatorowi w korzeniu, który podaje najbardziej prawdopodobną odpowiedź. Ponieważ klasyfikator jest z założenia słaby, nie można tej odpowiedzi całkowicie zaufać. Można jednak założyć, że klasyfikator wskaże właściwy podproblem (lub kilka z nich), gdzie, z dużym prawdopodobieństwem, znajduje się prawdziwa klasa danego przykładu. Dzieje się tak ze względu na, wspomnianą wyżej, własność skupiania się się podproblemów wokół dominujących klas. Jeśli więc klasyfikator w korzeniu wskaże *i*-tą klasę jako najbardziej prawdopodobną, należy sprawdzić, czy nie jest to inna klasa, która *należy* do tego samego podproblemu. Kluczowa będzie zdolność HCOC do budowy poprawnych podproblemów. Można pokazać, że tak jak dokładność słabego klasyfikatora pod względem wskazywania poprawnej klasy jest niska, tak jego dokładność pod względem wskazywania poprawnego podproblemu jest wysoka. Ta własność leży u podstaw poprawnego działania HCOC (Podolak, 2008; Podolak *et al.*, 2006).

Klasyfikator w korzeniu rozwiązuje w sposób przybliżony cały problem, tworząc podproblemy pierwszego rzędu. Algorytm uczący HCOC składa się z dwóch składowych: algorytmu nauczającego słaby klasyfikator umieszczony w korzeniu drzewa oraz algorytmu formującego podproblemy. Klasyfikatory w korzeniu mogą być właściwie dowolnego typu; jedynym wymaganiem jest, aby jako wynik zwracały one wektor prawdopodobieństw przynależności do klas $P(C|x)$. Z drugiej strony, algorytm tworzący podproblemy składające się z podzbiorów klas (będziemy je nazywali klastrami) bierze pod uwagę wyniki nauczania klasyfikatora. Na ich podstawie znajduje statystyczne zależności mówiące o tym, które klasy są do siebie podobne, a stąd często mylone, i powinny znaleźć się razem w jednym klastrze. Ta operacja jest powtarzana na każdym poziomie drzewa aż do osiągnięcia wystarczająco niskiego poziomu błędu.

HCOC, w trakcie klasyfikacji przykładu, znajduje dla każdego klasyfikatora w węźle potomnym jego oczekiwaną skuteczność dla podanego wektora atrybutów. Uzyskiwana jest w ten sposób miara „kompetencji" (odpowiedniości) klasyfikatorów w węzłach do klasyfikacji tego właśnie przykładu. Im jest wyższa, tym bardziej prawdopodobne, że podany przykład należy do rozpoznawanego przez klasyfikator podproblemu. Odpowiadające wagi pozwolą na składanie wyników,

które bierze pod uwagę sam klasyfikowany przykład. Wiele innych modeli klasyfikatorów złożonych ma te wagi ustalone na stałe. Nie są one obliczane osobno dla każdego klasyfikowanego przykładu i składanie wyników następuje przez proste głosowanie czy uśrednianie.

Zadania i plan pracy

W pracy, przed właściwą definicją klasyfikatora typu HCOC, pokazane będzie, że możliwe jest tworzenie podproblemów przez wybór klas, a także, na podstawie wyników nauczania klasyfikatora nadrzędnego, oszacowanie prawdopodobnej wysokości błędu dla jego podproblemów, a nawet ich symulacja. Pozwoli to ocenić, czy dany problem wraz z aktualnym jego modelem, tzn. zbudowanym dla niego klasyfikatorem, nadaje się do dalszego podziału w sensie HCOC, *przed* jego rzeczywistym wykonaniem. Dzięki temu możliwe będzie oszacowanie, czy warto jest dzielić aktualny problem z jego obecnym modelem, czy też należy go przebudować. Taka procedura jest sprawniejsza niż budowanie modelu monolitycznego wraz z pracochłonnym poszukiwaniem optymalnych parametrów.

Dodatkową zaletą takiego podejścia jest fakt, że kolejno tworzone problemy są coraz mniejsze, a wobec tego prostsze, ponieważ zawierają mniej wyjściowych klas. Pozwala to na ich uproszczenie i, prawdopodobnie, lepszą generalizację. Pokazane będzie jak rozdzielenie problemu na osobne grupy klas wpływa na ich entropię. Jednocześnie klasyfikator nadrzędny może dany przykład przypisać niepoprawnie do którejś z klas. Jeśli będzie ona należeć do jednego podproblemu z klasą prawdziwą, to możliwe będzie poprawienie pomyłki. Stąd bardzo ważnym elementem całego algorytmu nauczania jest algorytm klastrujący, który buduje podproblemy. Przy dodatkowym wysiłku możliwe jest zredukowanie wykorzystywanej w podproblemach przestrzeni atrybutów, gdyż nie wszystkie będą istotne dla aktualnego podproblemu (Frank *et al.*, 2003; Hastie *et al.*, 2001).

W następnym rozdziale przedstawiona będzie definicja modelu HCOC wraz z odniesieniami do teorii słabego nauczania. Ponieważ HCOC z założenia jest klasyfikatorem przydatnym dla rozpoznawania problemów, w których występuje wiele klas, potrzebna była modyfikacja definicji słabego nauczania i słabego klasyfikatora (ang. *weak*) dla takiego problemu. Dotychczas znane definicje nie były całkiem prawidłowe, a w każdym razie nie uwzględniały ważnych elementów klasyfikacji polegających na wyborze klasy wygrywającej. Nowa definicja bierze pod uwagę prawdopodobieństwo, że wartość aktywacji odpowiadająca prawidłowej klasie będzie największa w wektorze odpowiedzi $[P(C_1|x), \ldots, P(C_K|x)]$ klasyfikatora.

Następnie zademonstrowane zostaną założenia budowy wszystkich składników modelu HCOC. Zaproponowane będą różne modele nauczania poszczególnych składników, niektóre z nich jako adaptacje innych algorytmów nauczania maszynowego, inne całkowicie nowe. Będzie to miało zastosowanie w szczególności do zadań klastrowania i problemu tworzenia wag. Pokazane będzie, kie-

dy i dlaczego oczekiwana wartość funkcji kosztu proponowanego HCOC maleje wraz z liczbą dodawanych poziomów. Dowody będą się odnosić do założeń o budowie klasyfikatorów bazowych, w szczególności do wymagania, aby każdy z nich był co najmniej słaby.

W końcu zaprezentowane zostaną różne doświadczenia wykonane na ogólnie dostępnych danych. Opisane zostaną również zastosowania do szczególnych problemów, takich jak rozpoznawanie twarzy, rozpoznawanie obrazów i tekstur, problemów z teorii automatów.

Pokrewne do HCOC podejścia

Wielu autorów zwraca uwagę, że w problemach klasyfikacji wieloklasowej, zamiast budować duże monolityczne modele, lepiej jest składać szereg modeli prostszych, nawet binarnych (Kumar *et al.*, 1999; Giacinto, Roli, 2001; Duda *et al.*, 2000; Kittler *et al.*, 1997; Cesa-Bianchi *et al.*, 2006). Każdy z *lokalnych* modeli ma wtedy za zadanie rozwiązywać swoje własne zadanie, które jest podproblemem całego zadania. Podstawowymi kwestiami pozostającymi do rozwiązania są pytania: w jaki sposób generować podproblemy?, czy robić to przez podział przestrzeni atrybutów?, czy też może przez podział przestrzeni klas?, czy podproblemy powinny być rozłączne?, czy budowane modele powinny mieć z sobą związek?, czy podział ma służyć tylko do znalezienia jednego najbardziej właściwego lokalnego modelu czy też znalezienia ich grupy?, w jaki sposób składać wyniki?, i wiele innych. Tymi problemami zajmuje się poniższa praca o podejściu Klasyfikatora Hierarchicznego HCOC.

Wielu autorów proponuje różnego rodzaju podział problemu, chociaż jest to zwykle podział przestrzeni atrybutów. Poniżej przedstawione są niektóre z tych podejść, które są w pewien sposób pokrewne do HCOC. W dodatku A.5 zaprezentowane są inne algorytmy klasyfikatorów złożonych.

AdaBoost (m.in. Schapire, 1990; Schapire, Singer, 1999; Eibl, Pfeiffer, 2005) jest najbardziej znanym modelem boostingu, a więc modelu polegającego na budowaniu kolejnych klasyfikatorów bazowych tak, aby każdy kolejny zyskiwał na własnościach poprzednich. Przykładom uczącym x przypisywane jest prawdopodobieństwo, zgodnie z którym losowane są do nauczania kolejnych klasyfikatorów bazowych $Cl(i)$. Po nauczeniu k-tego klasyfikatora $Cl(k)$ sprawdzana jest poprawność klasyfikacji dla poszczególnych przykładów i te, które są rozpoznawane gorzej, będą częściej wybierane przy nauczaniu kolejnego klasyfikatora $Cl(k+1)$. Końcowa klasyfikacja przykładów tworzona jest jako suma ważona klasyfikacji przez poszczególne $Cl(k)$. Dzięki temu klasyfikator $Cl(k+1)$ skupia się w większym stopniu na przykładach niepoprawnie rozpoznawanych przez $Cl(k)$, a poszczególne klasyfikatory mogą być *słabe*. Proponowany algorytm HCOC też poprawia dotąd uzyskaną klasyfikację, jednak skupiając się na podproblemach definiowa-

nych jako grupy klas. Dzięki temu kolejne klasyfikatory rozpoznają problemy prostsze.

Feature-Subspace Aggregating (Feating) to model, który, na podstawie niektórych spośród atrybutów wejściowych, dzieli oryginalny problem poprzez enumeratywne generowanie rozdzielnych podprzestrzeni cech. Algorytm Feating tworzy drzewo wszystkich możliwych podziałów przestrzeni wejściowej, a dla każdej podprzestrzeni buduje w liściu lokalny model. Podobnie jak HCOC, podejście Feating kieruje się zasadą, że każdy taki lokalny model będzie zwykle dokładniejszy niż model globalny zbudowany z wykorzystaniem wszystkich przykładów. Dodatkowo, w wielu wygenerowanych podprzestrzeniach, niektóre atrybuty mają tylko niewiele wartości, co redukuje ich liczbę przy budowie lokalnego modelu (Ting *et al.*, 2010).

Poza przestrzeniami o bardzo małej liczbie atrybutów, liczba podprzestrzeni jest bardzo wysoka. Autorzy proponują więc losowe generowanie ustalonej liczby podprzestrzeni i budowę lokalnych modeli. W trakcie klasyfikacji danego przykładu algorytm ewaluuje wszystkie modele opisujące podprzestrzenie, do których przykład należy, a następnie wybiera klasyfikację przez głosowanie. Wydaje się, na co zwracają uwagę autorzy, że przydatne byłoby ważenie wybranych klasyfikatorów, jednak nie wiadomo, na jakiej podstawie miałyby być wybierane wagi. Najprostszym wyborem byłaby ogólna skuteczność każdego z klasyfikatorów, tak jak to się dzieje w podejściach typu *boosting* (Schapire, Singer, 1999). W modelu HCOC wagi zależą od klasyfikacji aktualnie rozpatrywanego przykładu przez modele w wyższych węzłach.

Inaczej niż w HCOC, w algorytmie Feating następuje podział przestrzeni atrybutów *bez* wykorzystania etykietowania przykładów. Ma to taką zaletę, że jest szybkie, jednak nie bierze pod uwagę własności algorytmów wykorzystywanych do budowy samych modeli, jak to się dzieje w HCOC.

Algorytm opisany w (Casasent, Wang, 2005) przypomina w niektórych założeniach HCOC. Autorzy wskazują na konieczność podziału zadania w problemach klasyfikacji wieloklasowej również dlatego, że mniejsze problemy są zwykle prostsze do rozwiązania. Zaproponowany algorytm buduje binarne drzewo poprzez podział zadania w każdym węźle na *rozłączne* grupy klas o tej samej liczności. Pozwala to na budowę drzewa o minimalnej głębokości. W każdym węźle wykorzystany jest algorytm SVRDM (Wang, Casasent, 2008), rozszerzenie algorytmu SVM, pozwalający na oddzielenie grupy klas w przestrzeni cech.

Podejście to jest w pewnym stopniu podobne do proponowanego w tej pracy modelu HCOC, ze względu na to, że buduje hierarchiczny model podziałów, stosując algorytmy działające w każdym węźle na podzbiorach przykładów należących do znalezionej grupy klas. Grupy klas ustalane są przy użyciu algorytmu klastrowania, podobnie jak w HCOC. W modelu HCOC klastrowana jest jednak przestrzeń wyjściowych klasyfikacji klasyfikatora ojca.

Takie podejście algorytmu HCOC modeluje przestrzeń klasyfikacji oraz niedokładności popełniane przez klasyfikator w węźle nadrzędnym.
Algorytm Wanga i Casasenta wyszukuje grupy klas, wykorzystując klastrowanie przez podział, a następnie w każdym węźle modeluje je przy wykorzystaniu modelu SVM. Model HCOC czyni to odwrotnie: najpierw usiłuje znaleźć przybliżenie rozkładu wieloklasowego, po czym dzieli przestrzeń rezultatów klasyfikacji przy wykorzystaniu klastrowania. Grupy klas (makroklasy w nomenklaturze Wanga i Casasenta, klastry w HCOC) są w opisanym podejściu rozłączne, podczas gdy w HCOC, dla zwiększenia dokładności, klastry mają wiele części wspólnych. Z tego względu model HCOC wykorzystuje w końcowej klasyfikacji liniową kombinację klasyfikacji wielu klasyfikatorów, a opisany model autorów idzie dokładnie jedną ścieżką drzewa. W nowszej pracy Wang, Casasent (2009) opisują rozmyte rozszerzenie tego algorytmu ewaluujące wszystkie ścieżki drzewa, w dalszym ciągu jednak rozłącznych podproblemów.

Map/Reduce jest modelem wzorowanym na rozwiązaniach pochodzących z języka Lisp. Jego celem jest maksymalnie łatwy podział zadania na wiele zadań prostszych w taki sposób, aby było możliwe rozproszenie ich rozwiązywania. Operacja *map* dla każdego rekordu danych wejściowych oblicza pojedynczy klucz, tzw. pośredni. Następnie operacja *reduce* wykorzystuje jedną (lub kilka bardzo zbliżonych) wartość klucza pośredniego i wszystkie związane z tą wartością rekordy, redukując zadanie do znacznie mniejszego. Dla każdej takiej grupy *reduce* oblicza wartość funkcji końcowej (będzie to zwykle co najwyżej jedna wartość). Te wartości są następnie scalane (Ibrahim *et al.*, 2009; Dean, Ghemawat, 2008; Nguyen *et al.*, 2010; Panda *et al.*, 2012).
Dla urównoleglenia operacje na grupach danych o tych samych wartościach klucza wykonywane są na wielu niezależnych komputerach. Jest kilka znanych implementacji modelu, np. MapReduce (Google) i HaDoop (Apache). Podejście MapReduce pozwala na znaczne przyspieszenie przetwarzania dzięki bardzo głębokiemu podziałowi i sprawnemu rozproszeniu. Z jednej strony model sprawdza się szczególnie dobrze, gdy zadanie wymaga przeglądnięcia bardzo dużej liczby danych przy jednokrotnym wykonaniu prostej operacji na każdym rekordzie. Z drugiej jednak strony problem jest od początku do końca rozwiązywany na poziomie pojedynczych rekordów. Rekordy złączone przez identyczną wartość klucza pośredniego nie tworzą w rzeczywistości nowego podproblemu. Jednocześnie efektywny podział na grupy rekordów jest w dużym stopniu uzależniony od zdefiniowanego przez projektanta wyrażenia dla klucza pośredniego. Potrzebna jest więc wiedza o oryginalnym problemie. Klasyfikator HCOC stara się natomiast podzielić zadanie na podproblemy, które mają podobny wpływ na rozwiązanie.

Drzewa słabych klasyfikatorów dla dopasowywania wzorców kształtów (Tsapanos *et al.*, 2011) zapamiętują w liściach wzorce kształtów, wykorzystując

w każdym węźle wewnętrznym, przypominający perceptron, słaby klasyfikator (podobnie jak HCOC). Decyduje on, czy w poszukiwaniu klasyfikacji należy przejść do lewego czy prawego poddrzewa. Dla każdego węzła tworzone są, w procesie nauczania gradientowego, macierze sterujące procesem decyzyjnym. Niepoprawna decyzja w jednym z węzłów może spowodować błędne dopasowanie. Ponieważ klasyfikatory są słabe, istnieje duże prawdopodobieństwo błędu. Z tego powodu proces rozpoznawania jest powtarzany wielokrotnie z odwracaniem decyzji w węzłach, w których przewaga jednego poddrzewa była niewielka. To rozwiązanie bardziej przypomina drzewa decyzyjne.

Rozdział 1

Podział zadania klasyfikacji

Wiele problemów może być skutecznie podzielonych na mniejsze części, stanowiących problemy tej samej klasy. Kolejne podziały tworzą podproblemy coraz prostsze do rozwiązania. Pozwala to na znalezienie rozwiązania dla każdej z części, a następnie złożenie tych rozwiązań, dając skuteczniejszy model (Hastie *et al.*, 2001; Bishop, 2006). Możliwe są dwa podstawowe podejścia:

- podział przestrzeni wejściowej atrybutów, odpowiedni podział zbioru uczącego, a następnie niezależne nauczanie na każdym z podzbiorów, tworząc cząstkowe hipotezy, na końcu złożone według ustalonej procedury,
- podział zbioru wyjściowych klas na podzbiory, podział zbioru uczącego tak, aby w każdym były przykłady z danego podzbioru klas, nauczanie dla każdego z tych podzbiorów uczących, tworząc hipotezy, w końcu składanie tych hipotez według ustalonej procedury.

Każde z tych podejść ma wiele możliwych wariantów. Rezultatem ich działania będzie jednak zawsze pewien klasyfikator złożony, ponieważ rozwiązanie zadania będzie się dzielić na części: znalezienie podzbiorów, nauczanie dla każdego z nich, a następnie łączenie wyników. Każdy z nich może mieć wiele poziomów.

Końcowy wynik nauczania będzie zależał od skuteczności algorytmu nauczania tworzącego poszczególne klasyfikatory oraz związane z nimi hipotezy, a także od sposobu ich składania. Przede wszystkim jednak od algorytmu, według którego podproblemy będą tworzone. Możliwe jest podejście, w którym podproblemy są znane przed rozpoczęciem nauczania, np. dzięki udziałowi eksperta w procesie nauczania. Jednak ekspert zwykle nie jest dostępny, nie ma też pewności co do jego nieomylności i umiejętności utworzenia skutecznego podziału. W większości przypadków konieczny będzie automatyczny proces tworzenia podproblemów.

Podstawowym celem systemów hierarchicznych jest podział całego zadania na podproblemy. W modelu HME (patrz dodatek A.5) przykłady uczące do poszczególnych sieci eksperckich są losowane z obszarów przestrzeni wejściowej określo-

nych przy wykorzystaniu założonego modelu, np. modeli gausowskich (wzorowane na podobieństwo modelu Expectation–Maximization). W HCOC tworzenie podproblemów następuje nie przez łączenie obszarów, lecz przez podział, przy czym nie jest on oparty na założonym *a priori* modelu probabilistycznym, ale na własnościach samego problemu. Te własności model HCOC znajduje poprzez próbę przybliżonego rozwiązania całości zadania, a następnie wyszukania podobieństw.

1.1. Klasyfikator

Definicja 1.1. Niech $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^{N}$ będzie skończonym zbiorem uczącym składającym się z N par takich, że $x \in \mathcal{X}$ jest wektorem atrybutów, a $c_i \in \mathcal{C} = \{C_1, \ldots, C_K\}$ jest prawidłową klasą pochodzącą ze skończonego zbioru. Klasyfikator Cl jest złożeniem $S \circ Cl$ takim, że

$$Cl \quad : \quad x \longrightarrow [0,1]^K, \tag{1.1}$$

$$S \quad : \quad [0,1]^K \longrightarrow \mathcal{C}. \tag{1.2}$$

Taka definicja pozwala na wyraźne oddzielenie procesu nauczania Cl, którego bezpośrednim wynikiem jest wektor prawdopodobieństw przynależności do klas, od procesu selekcji końcowej klasy realizowanej przez funkcję S. Najczęstszym wyborem S jest reguła arg max, polegająca na wyborze klasy C_i o największym prawdopodobieństwie przynależności, co jest zgodne z regułą *klasyfikatora Bayesa* żądającą, aby wybrać klasę najbardziej prawdopodobną. Oznacza to, że odpowiedź $Cl(x)$ w definicji 1.1 traktujemy jako wektor prawdopodobieństw przynależności do klas.

Rozdzielenie w definicji 1.1 na klasyfikator Cl i selektor S jest szczególnie istotne dla klasyfikatora HCOC, w którym poszczególne klasyfikatory składowe nie potrzebują wybierać końcowej klasyfikacji. Analiza wyników samego Cl pozwoli na ekstrakcję grup klas, które będą tworzyć podproblemy.

1.2. Podział przestrzeni klas

Alternatywnym do podziału przestrzeni atrybutów użytym w HCOC podejściem jest podział przestrzeni wyjściowej. Wybór ten można uzasadnić heurezą, że w zadaniu (reprezentowanym przez zbiór uczący) wyróżnione są klasy i zachodzi pomiędzy nimi podobieństwo.

Definicja 1.2. Klasy są „podobne“, jeśli podobne (w sensie zdefiniowanej odległości w przestrzeni aktywacji) są wzorce klasyfikacji obliczane przez dany klasyfikator (będzie tak zwykle, gdy wektory atrybutów opisujące przykłady z tych klas są też podobne).

W związku z tym dobrym sposobem może być grupowanie klas podobnych w podproblemy. Pozostaje pytanie, w jaki sposób to robić i jak można określić, *bez testowania*, na ile trudne do rozwiązania są poszczególne podproblemy. Jako miarę złożoności można wziąć oszacowanie miary ryzyka klasyfikatora rozwiązującego ten podproblem.

Hipoteza 1.3. *Niech $A \in \mathcal{A}$ będzie algorytmem nauczania maszynowego z klasy $\mathcal{A}$. Niech A buduje model dla K-klasowego problemu klasyfikacji danego przez zbiór uczący $\mathcal{D}$. $\mathcal{A}$ jest rodziną algorytmów o takiej samej lub zbliżonej złożoności obliczeniowej i pamięciowej.*

A tworzy hipotezę $h : \mathcal{X} \longrightarrow \mathcal{C}$, gdzie $\mathcal{C} = \{C_1, \ldots, C_K\}$. Niech $\mathcal{I}$ będzie podproblemem, będącym zadaniem znalezienia hipotezy $h_\mathcal{I} : \mathcal{X} \longrightarrow \mathcal{C}_\mathcal{I}$, gdzie $\mathcal{C}_\mathcal{I} \subsetneq \mathcal{C}$.

Problem znalezienia hipotezy $h_\mathcal{I}$ jest prostszy od znalezienia hipotezy h.

Dowód. Niech $D = \{(x_i, c_i), x_i \in \mathcal{X}, c_i \in \mathcal{C}\}_{i=1}^N$ będzie skończonym zbiorem danych uczących. Niech F będzie prawdziwą funkcją

$$F : \mathcal{X} \ni x \longrightarrow \xi(x), \tag{1.3}$$

gdzie rozkład

$$Dist(\xi(x)) = (\xi_1(x), \ldots, \xi_K(x)) \in [0,1]^K \tag{1.4}$$

jest wektorem przynależności x do klas C_i dla $i = 1, \ldots, K$. $\forall\ x\ \sum_{i=1}^K \xi_1(x) = 1$. Zmienna losowa $\xi(x)$ przyjmuje wartości w zbiorze etykiet $\mathcal{C}$. Zadanie modelowania funkcji F można zamienić na zadanie modelowania funkcji $\widetilde{F}$ takiej, że

$$\widetilde{F} : \mathcal{X} \ni x \longrightarrow Dist(\xi(x)) = (\xi_1(x), \ldots, \xi_K(x)) \in [0,1]^K. \tag{1.5}$$

Zadaniem algorytmu $A \in \mathcal{A}$ jest zbudowanie hipotezy (modelu) h będącej modelem funkcji $\widetilde{F}$

$$h : \mathcal{X} \ni x \longrightarrow h(x) \in [0,1]^K. \tag{1.6}$$

Niech $H_F(\mathcal{X})$ będzie entropią układu (problemu) F dla przestrzeni $\mathcal{X}$

$$H_F(X) = \int_\mathcal{X} H_F(x) p(x) dx, \tag{1.7}$$

$$H_F(x) = -\sum_{k=1}^K \log_K \xi_k(x) \xi_k(x). \tag{1.8}$$

Prawdziwy rozkład $p(x)$ nie jest znany, trzeba go wobec tego estymować ze zbioru uczącego D

$$\widehat{p}(x) = \frac{1}{N} \sum_{(x_i, c_i) \in D} K_B(x - x_i), \tag{1.9}$$

gdzie K_B jest jądrowym estymatorem gęstości dla pasma o szerokości B. Empiryczna wartość $\hat{H}$ ma wobec tego postać

$$\widehat{H}(X) = \frac{1}{N} \sum_{x \in X} \hat{H}(x), \tag{1.10}$$

$$\widehat{H}(x) = -\sum_{k=1}^{K} \hat{p}_k(x) \log_K \hat{p}_k(x), \tag{1.11}$$

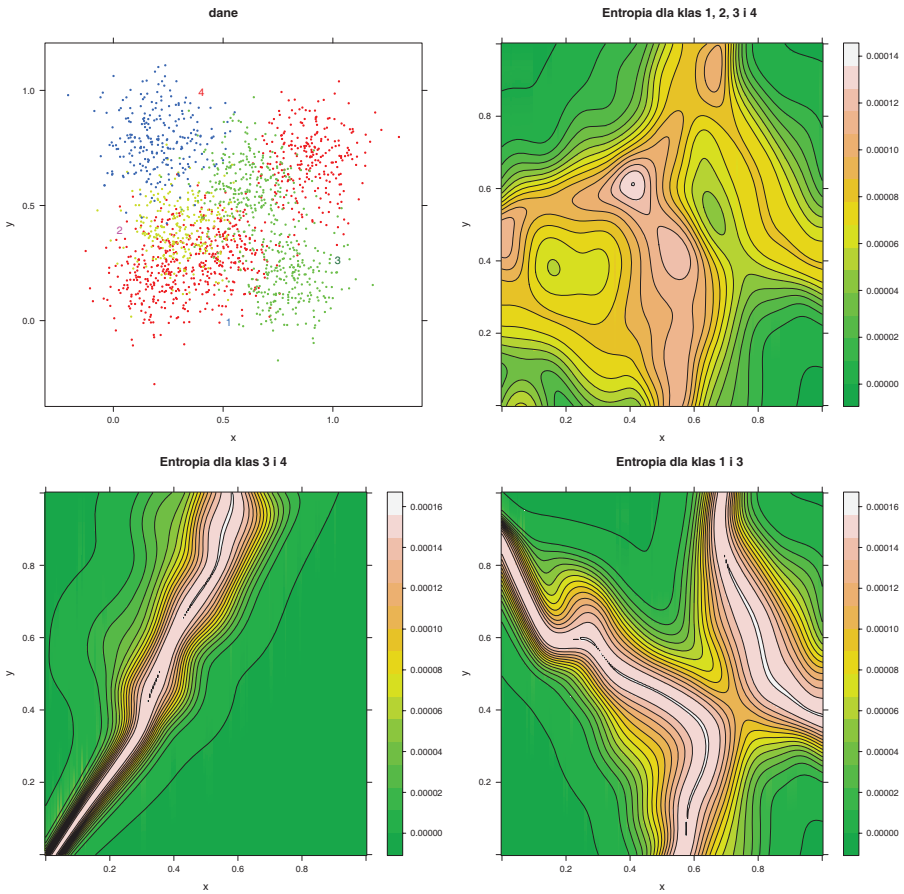
$$\widehat{p}_k(x) = \frac{\sum_{(x_i,c_i) \in D : c_i = C_k} K_B(x_i - x)}{\sum_{(x_i,c_i) \in D} K_B(x_i - x)}. \tag{1.12}$$

Miara entropii (1.10), przy wykorzystaniu (1.11) i ewaluacji gęstości za pomocą jądrowego estymatora gęstości klas, będzie dla podproblemu składającego się z podzbioru klas oryginalnego problemu mniejsza niż dla oryginalnego problemu. Miara entropii, tak jak może posłużyć do oceny istotności atrybutów (Duch *et al.*, 2004a), tak też można jej użyć do oceny różnicy w złożoności problemów przy ich redukcji o poszczególne klasy. W (1.10) wartość liczby klas K przy pomiarze dla całego problemu i podproblemu musi być identyczna, aby wartości były porównywalne. Podstawa logarytmu w definicji entropii może być dowolną wartością większą od 1. Pozostawienie tej samej podstawy K odpowiada teoretycznej sytuacji, w której klasyfikator rozwiązuje problem o takiej samej liczbie klas, jednak dla części z nich nie są podane przykłady. Odpowiada to wyborowi grupy klas i rozwiązywania tego jak oryginalnego problemu. Intuicyjnie taki właśnie problem jest prostszy, co jest zgodne z jego mniejszą entropią.

Alternatywnie do entropii krzyżowej (1.11) można wykorzystać, na przykład, indeks Giniego $G(x) = \sum_{k=1}^{K} \widehat{p}_k(1 - \widehat{p}_k)$ albo nawet błąd potencjalnej klasyfikacji $1 - \widehat{p}_k$. Miary (1.11) i indeks Giniego są różniczkowalne i lepiej oddają zmiany klasyfikacji (Hastie *et al.*, 2001).

Na rysunku 1.1 pokazany jest przykładowy zbiór danych oraz mapy entropii. Cały zbiór danych składa się z 750 przykładów z klasy 1, 250 przykładów z klasy 2, 500 z klasy 3 i 250 z klasy 4. Dane zostały wylosowane z 7 chmur gausowskich, odpowiednio trzech, jednej, dwóch i jednej po 250 punktów każda, dla kolejnych klas. Z wykorzystaniem jądrowego estymatora gęstości ewaluowane zostały gęstości prawdopodobieństwa dla każdej z klas i wartość zdefiniowanej entropii z wzoru (1.10). Przybliżone wartości wynosiły odpowiednio: dla wszystkich czterech klas $\widehat{H}(X_{1,2,3,4}) = 0.565$, dla klas 3 i 4: $\widehat{H}(X_{3,4}) = 0.127$, dla klas 1 i 3 $\widehat{H}(X_{1,3}) = 0.349$.

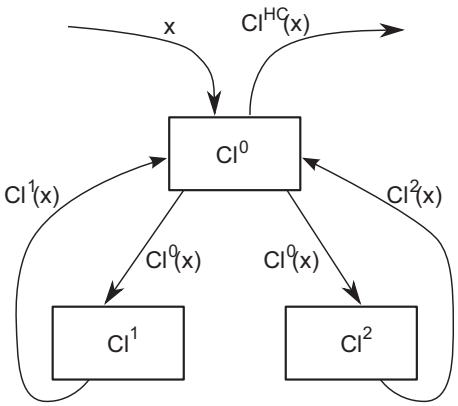
Rysunek dla klas 3 (zielone punkty) i 4 (niebieskie) wyraźnie obrazuje, że ten problem jest stosunkowo prosty i klasy dzielą płaszczyznę na dwie części. Rysunek dla klas 1 (czerwone) i 3 (zielone) demonstruje problem, gdzie klasy przecinają się, i największy problem pojawia się właśnie w miejscu przecięcia. Mapa entropii dla wszystkich klas wyraźnie pokazuje, gdzie problem decyzyjny jest największy: jest to miejsce, gdzie przecinają się przykłady z wszystkich klas w okolicach punktu $(0.4, 0.6)$. Wartość entropii ma tam maksimum. □



Rysunek 1.1. Empiryczne mapy entropii dla zbioru danych. Od lewej do prawej: dane, entropia dla wszystkich klas, dla klas 3 (zielone punkty na rysunku z danymi) i 4 (niebieskie), dla klas 1 (czerwone) i 3 (zielone). Obszary zielone odpowiadają najniższej, a białe najwyższej lokalnej entropii

Teoretycznie można wykorzystać zaproponowaną miarę dla oceny złożoności podproblemów. Nie będzie to jednak efektywne, gdyż sama postać entropii nie daje wskazówek, które z klas należy usunąć, aby osiągnąć największy jej spadek. Podobne rozumowanie i procedura wykorzystywane są przy budowie drzew decyzyjnych dla znalezienia podziału problemu na dwa rozdzielne podproblemy. Tam jednak analizowane są testy pojedynczych atrybutów i nie ma mowy o większej liczbie nakładających się podproblemów.

W proponowanym w pracy klasyfikatorze HCOC nauczanie działa naprzemiennie z zadaniem podziału na podzadania. Przepływ danych w uproszczonym HCOC pokazany jest na rysunku 1.2.



Rysunek 1.2. Przepływ danych w maksymalnie uproszczonym HCOC

Przykład reprezentowany przez wektor atrybutów x jest podawany na wejściu klasyfikatora Cl^0 w korzeniu modelu HCOC. Cl^0 z założenia rozpoznaje wszystkie klasy w całym problemie (grupa klas oznaczona jako Q^0), czyni to jednak w sposób przybliżony – jest klasyfikatorem *słabym* (ang. *weak*). Jak to jest pokazane w dyskusji w rozdziale 2.1, w klasyfikatorach słabych przykłady będą się grupować wokół przykładów reprezentujących często występujące klasy (Podolak, Roman, 2011a, 2012). W związku z tym istnieje duże prawdopodobieństwo, że jeśli nawet odpowiedź Cl^0 będzie nieprawidłowa, to wektor wejściowy x zaliczony będzie do grupy zawierającej prawidłową odpowiedź. Model HCOC budowany jest w dwóch przeplatających się cyklach: budowy klasyfikatorów bazowych i podziału na grupy odpowiadające podproblemom. Podział będzie wykonywany tak, aby maksymalizować prawdopodobieństwo, że klasyfikowany x zostanie przekazany do grupy zawierającej prawdziwą klasę.

W takiej sytuacji możliwe jest oszacowanie prawdopodobieństwa, że prawidłowa klasa znajduje się w grupach klas (zwanych dalej klastrami) Q^1 bądź Q^2, które są *podproblemami* Q^0, osobno rozwiązywanymi przez klasyfikatory, odpowiednio, Cl^1 i Cl^2. W końcu wyniki będą złączone, biorąc pod uwagę prawdopodobieństwa klastrów.

Zadanie podziału oryginalnego problemu Q^0 na podproblemy zostanie w modelu HCOC wykonane przy wykorzystaniu innych narzędzi do klastrowania danych, przy czym potrzebna będzie ocena, który z podziałów będzie dawać lepsze wyniki. Najprostszym podejściem będzie zbudowanie HCOC i zmierzenie wartości ryzyka $R_{emp}[HCOC]$ dla każdego podziału. To rozwiązanie jest w oczywisty sposób kosztowne. Poniższa hipoteza uzasadnia wykorzystanie wyników znanego algorytmu nauczania do oszacowania wartości ryzyka dla podproblemu.

Hipoteza 1.4. *Niech A będzie algorytmem nauczania maszynowego, za pomocą którego budowany jest model K-klasowego problemu klasyfikacji. A tworzy hipotezę* $h : \mathcal{X} \longrightarrow \mathcal{C}$,

gdzie $\mathcal{X}$ jest przestrzenią atrybutów, a $\mathcal{C} = \{C_1, \ldots, C_K\}$ jest zbiorem etykiet dla K klas. Niech $\mathcal{I}$ będzie podproblemem, będącym zadaniem znalezienia hipotezy $h_\mathcal{I} : \mathcal{X}_\mathcal{I} \longrightarrow \mathcal{C}_\mathcal{I}$, gdzie $\mathcal{C}_\mathcal{I} \subsetneq \mathcal{C}$.

Jeśli do zbudowania $h_\mathcal{I}$ dla podproblemu wykorzystany będzie ten sam algorytm A, to na podstawie ryzyka $R[h]$ można oszacować ryzyko $R[H_\mathcal{I}]$.

W kolejnych podrozdziałach zawarte jest uzasadnienie tej hipotezy.

Wykorzystując tę hipotezę, możemy wybrać wiele podproblemów, oszacować ich ryzyko, wyszukując te, które pozwolą na zbudowanie najlepszego klasyfikatora HCOC. Ułatwi i przyspieszy to budowę całego modelu.

1.2.1. Losowa symulacja klasyfikatora dla podproblemu

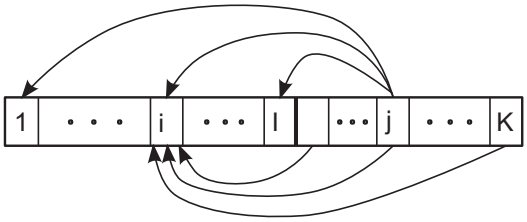
Jeśli algorytm nauczania z rodziny algorytmów $\mathcal{A}$ jest wykorzystywany do tworzenia obydwu hipotez $h()$ oraz $h_\mathcal{I}()$, w szczególności takich, że $\mathcal{C}_\mathcal{I} \subsetneq \mathcal{C}$, to obydwie hipotezy będą miały wspólne cechy. W szczególności, obydwie hipotezy powinny dawać zbliżone wyniki klasyfikacji w obszarze występowania klas z $\mathcal{C}_\mathcal{I}$, przy czym $h_\mathcal{I}$ będzie dokładniejsza w tym obszarze, ponieważ jej błąd poza nim jest z definicji równy 0. Algorytm A, budując $h_\mathcal{I}()$, nie potrzebuje więc minimalizować błędu poza $\mathcal{C}_\mathcal{I}$, a stąd $h_\mathcal{I}()$ jest w $\mathcal{C}_\mathcal{I}$ co najmniej tak dobra jak hipoteza $h()$. Z drugiej strony, jeśli x nie należy do $\mathcal{C}_\mathcal{I}$, to żadna z tych hipotez nie podaje prawidłowej odpowiedzi w ramach podproblemu $\mathcal{I}$.

Niech klasyfikator Cl realizuje hipotezę h. Dla przykładów z $\mathcal{C}_\mathcal{I}$ odpowiedź klasyfikatora $Cl_\mathcal{I}$ będzie zbliżona do odpowiedzi Cl. Niech Cl będzie znany. Można zaproponować implementację hipotezy $h_\mathcal{I}()$, w której metoda wykorzystuje odpowiedź Cl, jeśli należy ona do $C_\mathcal{I}$; w przeciwnym wypadku wybierana jest losowa klasa z tego podproblemu. Takie rozwiązanie opisane jest na rysunku 1.3.

Input: $x \in \mathcal{C}_\mathcal{I}$ – wektor atrybutów do klasyfikacji
Input: Cl – algorytm klasyfikacji rozwiązujący problemy dla $\mathcal{C} \supsetneq \mathcal{C}_\mathcal{I}$
Output: klasyfikacja x
$j \longleftarrow \arg\max_i Cl_i(x)$
if $j \in \mathcal{I}$ **then**
  return j
else
  return $random(\mathcal{I})$
end

Rysunek 1.3. Budowa hipotezy dla podproblemu przez losowy wybór odpowiedzi

Wszystkie przykłady prawidłowo zaklasyfikowane przez Cl są w dalszym ciągu klasyfikowane poprawnie. Te źle klasyfikowane mają prawdopodobieństwo $1/|\mathcal{I}|$ prawidłowej klasyfikacji. Jest to zobrazowane na rysunku 1.4.



Rysunek 1.4. Przypisanie do klas w $\mathcal{I} = \{C_1, \ldots, C_{|\mathcal{I}|}\}$. Przykłady zaklasyfikowane przez Cl do różnych klas w całym problemie mogą być przypisane do klasy C_i, należącej do podproblemu $\mathcal{I}$ (dolne strzałki), bądź też różne przykłady przypisane przez Cl do C_j mogą być przez algorytm przeniesione do różnych klas wewnątrz $\mathcal{I}$. Dla czytelności klasy z podproblemu $\mathcal{I}$ są indeksowane od 1 do $|\mathcal{I}|$

Definicja 1.5. Niech Cl będzie klasyfikatorem dla K-klasowego problemu. Macierz $M = \{m_{ij}\}_{i,j=1}^{K}$ nazywamy macierzą niepoprawnych klasyfikacji (pomyłek, ang. *misclassification* lub *confusion matrix* (Parker, 2001)), jeśli $m_{ij} = P(pr = C_j | tr = C_i, Cl)$, tzn. odpowiada prawdopodobieństwu, że Cl zaklasyfikuje element x z prawdziwej (*tr* jak *true*) klasy C_i jako należący do klasy C_j (*pr* jak *predicted*).

Macierz M jest przybliżeniem wartości ryzyka (ang. *risk*), w szczególności

$$R[Cl] = 1 - \sum_{i=1}^{K} p_i m_{ii}, \tag{1.13}$$

co przenosi ewaluację $R[Cl]$ ze zorientowanej na przykłady na zorientowaną na klasy przykładów. Zakładamy, że macierz M jest stochastyczna.

Można dla $Cl_{\mathcal{I}}$ skonstruować macierz niepoprawnych klasyfikacji $\widehat{M}$

$$\widehat{m}_{ik} = m_{ik} + \sum_{j \notin \mathcal{I}} m_{ij}, \tag{1.14}$$

gdzie m_{ij} są elementami macierzy dla Cl. Macierz $\widehat{M}$ opisuje działanie algorytmu 1.3.

Własność 1.6. *Macierz $\widehat{M}$ jest stochastyczna.*

Dowód. Dla dowolnego $i \in \mathcal{I}$

$$\sum_{k \in \mathcal{I}} \widehat{m}_{ik} = \sum_{k \in \mathcal{I}} m_{ik} + \frac{1}{|\mathcal{I}|} \sum_{k \in \mathcal{I}} \sum_{j \notin \mathcal{I}} m_{ij} = \sum_{k \in \mathcal{I}} m_{ik} + \sum_{j \notin \mathcal{I}} m_{ij} = 1. \tag{1.15}$$

Całe $\widehat{M}$ sumuje się więc do $|\mathcal{I}|$. □

1.2.2. Symulacja z wykorzystaniem prawa Bayesa

W założeniach HCOC leży fakt, iż błędna klasyfikacja też niesie z sobą informację. Jeśli przykład z prawdziwej klasy C_i został *błędnie* zaklasyfikowany jako C_j, wtedy prawdopodobieństwo $P(tr = C_i|pr = C_j)$ powinno to odzwierciedlać. Należy więc przy konstrukcji klasyfikatora $Cl_{\mathcal{I}}$ uwzględnić zarówno klasyfikację Cl_j dla $j \in \mathcal{I}$, jak i prawdopodobieństwa $P(tr = C_i|pr = C_j)$, które można wyliczyć, korzystając z prawa Bayesa

$$P(tr = C_k|pr = C_j) = \frac{m_{kj}P(C_k)}{\sum_{t=1}^{K} m_{tj}P(C_t)}, \tag{1.16}$$

gdzie $P(C_k)$ jest prawdopodobieństwem *a priori* wystąpienia przykładu z klasy C_k. To podejście pokazane jest na rysunku 1.5. Widać to także na rysunku 1.4: na końcowy wybór klasy C_i (dolne strzałki) wpływa zarówno wybór przez Cl klasy C_i, jak też prawdopodobieństwa wyboru innych klas. Jaki jest ten wpływ, wynika ze związku C_i z innymi klasami, a ten związek będzie możliwy do określenia przez analizę macierzy niepoprawnych klasyfikacji M. Analiza ta nie wymaga dodatkowego nauczania. Rozkład wartości w M pokazuje nie tylko związki między klasami, ale także wpływ, jaki mają one na aktualnie używany klasyfikator.

Input: $x \in \mathcal{C}_{\mathcal{I}}$ – wektor atrybutów do klasyfikacji
Input: Cl – algorytm klasyfikacji rozwiązujący problemy dla $\mathcal{C} \supsetneq \mathcal{C}_{\mathcal{I}}$
Output: klasyfikacja x
$j \longleftarrow \arg\max_i Cl_i(x)$
if $j \in \mathcal{I}$ **then**
  return j
else
  return $\arg\max_{i \in \mathcal{I}} P(tr = C_i|pr = C_j)$
end

Rysunek 1.5. Ewaluacja klasyfikatora dla podproblemu z wykorzystaniem prawdopodobieństwa *a posteriori* i prawa Bayesa

---

Odpowiadająca algorytmowi z 1.5 macierz $\overline{M}$ będzie miała postać

$$\overline{m}_{ik} = m_{ik} + \sum_{j \notin \mathcal{I}}^{K} m_{ij}P(tr = k|pr = j) = m_{ik} + \sum_{j \notin \mathcal{I}}^{K} m_{ij}\frac{m_{kj}P(C_k)}{\sum_{t=1}^{K} m_{tj}P(C_t)}. \tag{1.17}$$

$\overline{M}$ *nie* jest stochastyczna, ponieważ wzór (1.17) dla dowolnego $j \notin \mathcal{I}$ wykorzystuje tylko elementy $k \in \mathcal{I}$ oraz $P(tr = C_k|pr = C_j)$, w związku z czym wiersze $\overline{M}$ nie sumują się do jedności.

Prawidłową postacią jest odpowiednio przeskalowana macierz $\widetilde{M}$

$$\widetilde{m}_{ik} = m_{ik} + \sum_{j \notin \mathcal{I}} m_{ij} \frac{P(tr = C_k | pr = j)}{\sum_{l \in \mathcal{I}} P(tr = l | pr = j)}. \tag{1.18}$$

Własność 1.7. *$\widetilde{M}$ jest stochastyczna.*

Dowód. Dla dowolnego $i \in \mathcal{I}$

$$\begin{aligned} \sum_{k \in \mathcal{I}} \widetilde{m}_{ik} &= \sum_{k \in \mathcal{I}} m_{ik} + \sum_{k \in \mathcal{I}} \sum_{j \notin \mathcal{I}} m_{ij} \frac{P(k|j)}{\sum_{l \in \mathcal{I}} P(l|j)} \\ &= \sum_{k \in \mathcal{I}} m_{ik} + \sum_{j \notin \mathcal{I}} m_{ij} \sum_{k \in \mathcal{I}} \frac{P(k|j)}{\sum_{l \in \mathcal{I}} P(l|j)} = 1. \end{aligned} \tag{1.19}$$

□

Mając nauczony klasyfikator Cl, możemy wyliczyć ryzyko dla podproblemu $\mathcal{I}$ jako

$$R[Cl_{\mathcal{I}}] = 1 - \sum_{i \in \mathcal{I}} \widetilde{m}_{ii} \frac{P(C_i)}{\sum_{k \in \mathcal{I}} P(C_k)}. \tag{1.20}$$

Wystarczy wyliczyć z wzoru (1.18) elementy leżące na przekątnej $\widetilde{M}$. Jedyną uciążliwością jest konieczność znalezienia wartości prawdopodobieństwa *a priori* dla klas w $\mathcal{I}$. Ponieważ nie ma innej informacji poza zbiorem uczącym, trzeba te wartości oszacować właśnie ze zbioru $\mathcal{D}$, zakładając, że jest reprezentatywny dla problemu.

Doświadczenie 1.8. *Dla doświadczalnego porównania nauczone zostały proste klasyfikatory dla znanych problemów vowel i primary tumor (Newman* et al.*, 1998; Zwitter, Sokolic, 1988). Problem vowel ma 990 przykładów dla 11 klas wyjściowych, dokładnie po 90 przykładów dla każdej klasy. Natomiast problem primary tumor ma 339 przykładów dla 21 nierównolicznych klas*[1]*. Problem primary tumor jest bardziej złożony również ze względu na nierównomierność dostępności przykładów dla różnych klas. Dodatkowe testy wykonane zostały dla innych popularnych zbiorów audiology i arrhythmia.*

W doświadczeniu nauczonych zostało po 10 klasyfikatorów rozwiązujących cały problem, a następnie generowane były wszystkie podproblemy o liczności od 2 do $K-1$, gdzie K jest liczbą klas[2]*. Dla każdego podproblemu ewaluowane było ryzyko hipotezy, wykorzystując wzory (1.18) oraz (1.20) i wykorzystując wyniki nauczania 10 klasyfikatorów bazowych, co dawało przedział wartości $R[Cl_{\mathcal{I}}]$ ewaluowanych z różnych nauczonych sieci. Szerokość tego interwału w dużym stopniu zależy od stopnia złożoności aktualnie rozwiązywanego podproblemu i użytej w tym celu sieci.*

[1] W oryginalnym zbiorze uczącym są 22 klasy, jednak dla jednej z nich nie ma żadnego przykładu.

[2] Ponieważ w problemie primary tumor była bardzo duża liczba podproblemów, dla każdej wielkości podproblemów losowanych było 5000 przykładów dla wykonania testów.

Następnie, dla każdego podproblemu nauczone było osobnych 5 (dla pewnego uniezależnienia się od wartości początkowych) klasyfikatorów. To dało możliwość porównania ewaluacji z wartościami rzeczywistymi. Do symulacji wykorzystane zostały sieci neuronowe osobno dla dwóch i trzech neuronów w warstwie ukrytej. Sieci neuronowe pozwalają w prosty sposób sterować poprzez wielkość warstwy ukrytej złożonością danego modelu.

Wykresy korelacji wyliczonych wartości ryzyka dla podproblemu z wartościami znalezionymi z eksperymentów są pokazane na rysunkach 1.6 i 1.7. Dla czytelności wyniki zostały posortowane według wartości ewaluacji funkcji ryzyka. Rzeczywiste wartości funkcji ryzyka zostały pokazane jako drobne punkty. Wartości ryzyka, pochodzące z uśrednienia pięciu sieci neuronowych, są w dużym stopniu uzależnione od losowania początkowych wartości wag, stąd dla wszystkich ewaluacji ryzyka podproblemu jest ona wartością pesymistyczną. Ważna jest wysoka korelacja między ewaluacją a rzeczywistą wartością $R[Cl_{\mathcal{I}}]$ potwierdzająca wiarygodność tych ewaluacji. Korelacje pomiędzy wartościami funkcji ryzyka ewaluowanymi z niezależnie nauczonych dla każdego podproblemu 20 sieci a ewaluacjami na podstawie wzoru (1.20) pokazane są w tabeli 1.1.

Wartości korelacji zależą także od rozmiarów podproblemów, co obrazuje tabela 1.2.

Tabela 1.1. Korelacja ρ Spearmana pomiędzy wartościami ryzyka dla podproblemów wyliczonymi z symulacji a wartościami obliczonymi z wzoru (1.20) dla niektórych problemów. Kolejne kolumny odpowiadają różnym liczbom neuronów ukrytych użytych w sieciach

Problem	2	błąd	3	błąd	5	błąd
Vowel	0.934	0.001	0.960	0.009	0.958	0.016
Tumor	0.852	0.024	0.844	0.035	0.845	0.032
Audiology	0.922	0.011	0.893	0.007	0.873	0.020
Arrhythmia	0.992	0.001	0.986	0.001	0.967	0.002

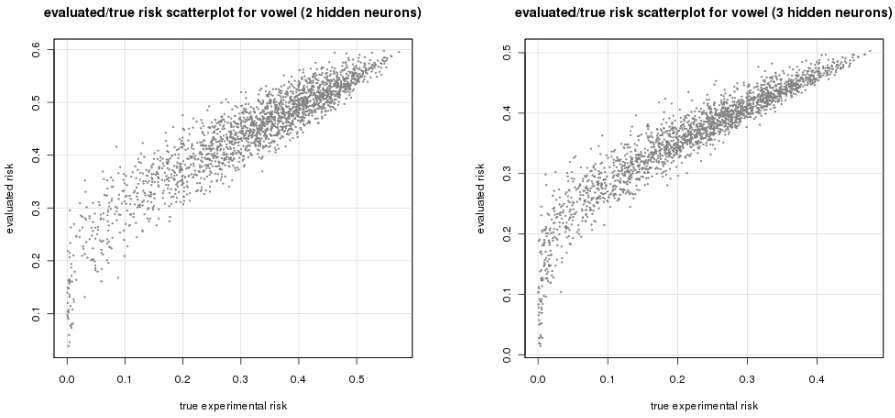
1.2.3. Wykorzystanie ewaluacji ryzyka w rzeczywistym problemie

W modelu HCOC konieczne jest nauczenie klasyfikatora bazowego dla oryginalnego problemu danego przez $\mathcal{D}$, a następnie podzielenie go na podproblemy odpowiadające różnym podzbiorom klas $\mathcal{I}_l,\ l = 1, \ldots, L$. Nauczanie wykorzystuje przyjęty algorytm uczący, który jest oparty na minimalizacji oczekiwanej wartości funkcji ryzyka $R[Cl]$. Po tym etapie wykorzystywany jest inny algorytm dokonujący podziału (Podolak, 2008; Podolak, Roman, 2011a) (algorytmy klastrowania są omówione szczegółowo w rozdziale 2.6).

Algorytm klastrujący powinien mieć możliwość oceny skuteczności danego podziału, tak aby można było wybrać najlepszy. W zasadzie jest to możliwe *jedy-*

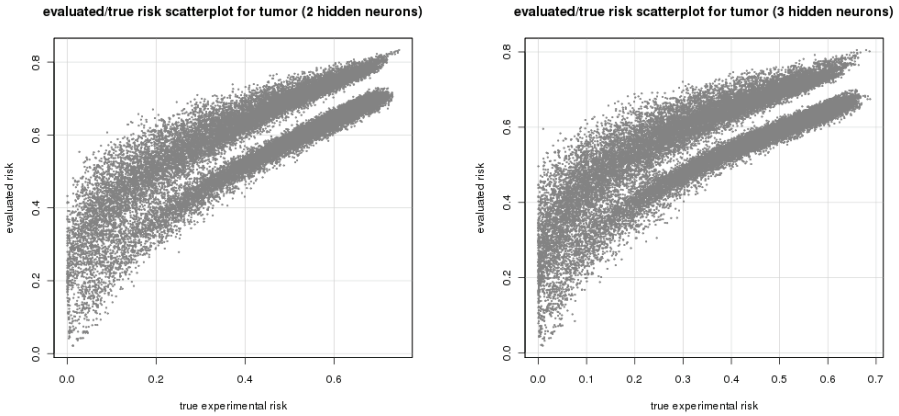
Tabela 1.2. Korelacje ρ Spearmana dla błędów podproblemów problemu tumor i ich ewaluacji. Pokazane są wszystkie rozmiary podproblemów od 2 do 20 (spośród 21 klas). Korelacja dla wszystkich przykładów wynosiła $\rho = 0.852$. W doświadczeniu użyto sieci warstwowych z 2 neuronami ukrytymi

Rozmiar	ρ	Błąd	Rozmiar	ρ	Błąd
2	0.174	0.0367	12	0.623	0.0103
3	0.580	0.0577	13	0.587	0.0083
4	0.649	0.0632	14	0.576	0.0060
5	0.676	0.0557	15	0.535	0.0044
6	0.671	0.0490	16	0.585	0.0030
7	0.673	0.0401	17	0.547	0.0020
8	0.662	0.0321	18	0.606	0.0013
9	0.670	0.0245	19	0.681	0.0007
10	0.638	0.0194	20	0.788	0.0003
11	0.648	0.0144			



Rysunek 1.6. Ewaluacja wartości funkcji ryzyka podproblemów problemu vowel. Jako nadrzędnego klasyfikatora użyto sieci neuronowej o 2 (lewy rysunek) i 3 neuronach ukrytych (prawy rysunek). Więcej ukrytych neuronów daje mniejszy przedział ewaluacji

nie przez nauczenie wynikających z tego klasyfikatorów złożonych, co jest oczywiście bardzo kosztowne. O wiele mniej kosztownym (ale i mniej dokładnym) rozwiązaniem może być ewaluacja wartości ryzyka, wykorzystując przedstawione podejście poprzez ewaluację macierzy $\widetilde{M}$ i wybór najlepszego klastrowania.



Rysunek 1.7. Ewaluacja wartości funkcji ryzyka podproblemów problemu vowel. Jako nadrzędnego klasyfikatora użyto sieci neuronowej o dwóch (lewy rysunek) i trzech neuronach ukrytych (prawy rysunek)

Przykład 1.9. *Wykorzystując funkcję kosztu* $\ell(x, C_{tr(x)}, Cl(x)) = 1 - Cl_{tr(x)}(x)$*, gdzie* $tr(x)$ *jest indeksem poprawnej klasy,* $C_{tr(x)}$ *poprawną klasą, a* $Cl_{tr(x)}(x)$ *wartością aktywacji dla poprawnej klasy przykładu* x*, można oszacować z góry ryzyko dla dwupoziomowego klasyfikatora typu HCOC (dokładne wyliczenia w rozdziale 2.7.1) jako*

$$R[HCOC] = 1 - \frac{1}{\max\limits_{k'} \sum_{l'=1}^{L} f_{k'l'}} \sum_{i=1}^{K} p_i \sum_{l=1}^{L} \sum_{k=1}^{K} f_{kl} m_{ii}^{l} m_{ik}^{0}, \tag{1.21}$$

gdzie m_{ik}^{0} *są elementami macierzy pomyłek (*misclassification*) dla klasyfikatora* Cl^0 *w korzeniu,* f_{kl} *są elementami zero-jedynkowej macierzy klastrowania* F *o wymiarach* $K \times L$ *(*K *jest liczbą oryginalnych klas, a* L *liczbą podproblemów),* p_i *są prawdopodobieństwami* a priori *klas.*

W tym wzorze występują także m_{ii}^{l}*, które są elementami przekątnych macierzy* M^l *dla nienauczonych jeszcze klasyfikatorów w kolejnym poziomie. Przyjmując* $M^l = \widetilde{M}^l$*, gdzie* $\widetilde{M}^l$ *jest ewaluacją macierzy pomyłek z wzoru (1.18) dla podproblemu złożonego z klas w klastrze* Q^l*, można wykorzystać wzór (1.21) jako funkcję kosztu algorytmu klastrowania. W rozdziale 2.6 podane są dokładne wyliczenia.*

Takie podejście pozwoli szybko nauczyć więcej niż jeden klasyfikator Cl^0 *w korzeniu i wybrać ten, który daje najlepsze wyniki. Jest to o tyle istotne, że niekoniecznie klasyfikator w korzeniu o najniższym ryzyku da najlepszy klasyfikator złożony. Na przykład, niech bazowy Cl będzie zbudowany na podstawie o tzw. zero-rule, a więc wybór odpowiedzi zawsze tej klasy C, która jest najliczniejsza. Predykcja Cl jest więc niezależna od samego wektora atrybutów x i dla wystarczająco licznej w zbiorze uczącym klasy C będzie dawać*

niskie $R[Cl]$. Taki klasyfikator nie będzie jednak dobrym punktem wyjścia do podziału na podproblemy.

Doświadczenie 1.10. *Dla pokazania możliwości podejścia do nauczania przy wykorzystaniu oszacowania błędów klasyfikatorów pochodnych wykonany został eksperyment polegający na ewaluacji wartości $R[HCOC]$ dla dwupoziomowego HCOC. Wartość ryzyka $R[HCOC]$ była obliczana na cztery sposoby:*

A *po nauczeniu korzenia Cl^0 i jego potomków Cl^i, wartość $R[HCOC]$ została obliczona na podstawie przykładów; odpowiada to pełnemu nauczeniu HCOC i policzeniu jego błędu z przykładów,*

B *wykorzystując ewaluację $R[HCOC]$ z wzoru (1.21) dla nauczonego Cl^0 oraz macierzy pomyłek M^l obliczonych dla nauczonych klasyfikatorów potomnych Cl^l; odpowiada to nauczeniu pełnego HCOC, ale ewaluacji $R[HCOC]$ z wzoru (1.21),*

C *$R[HCOC]$ obliczone, wykorzystując ewaluację błędu HCOC z wzoru (1.21) z Cl^0, nauczonym i klastrowanym, a następnie ewaluowanych na podstawie wzoru (1.18) przekątnych macierzy pomyłek $\widetilde{M}^l$; odpowiada to ewaluacji HCOC jedynie na podstawie nauczonego Cl^0,*

D *$R[HCOC]$ obliczone z danych: po nauczeniu Cl^0 i klastrowaniu, klasyfikatory potomne Cl^l symulowane z wykorzystaniem algorytmu z rysunku 1.5; odpowiada to analogicznej jak w punkcie **C** ewaluacji, ale wykorzystując algorytm.*

Tabela 1.3. Wartości korelacji Spearmana ρ dla różnych ewaluacji $R[HCOC]$ dla funkcji kosztu $\ell(x, C_{true(x)}, Cl(x)) = 1 - Cl_{true(x)}(x)$. Wyniki pochodzą z uśrednienia 12 000 architektur, z wykorzystaniem sieci neuronowych z różną liczbą neuronów ukrytych, uczonych różną liczbę epok (od słabych do silnych). Dla każdego nauczonego klasyfikatora w korzeniu nauczonych było 25 różnych zestawów klasyfikatorów potomnych

	Tumor	Vowel	Audiology	Arrhythmia
A–B	0.864	0.753	0.751	0.935
A–C	0.843	0.690	0.803	0.870
A–D	0.851	0.712	0.857	0.944
B–C	0.974	0.807	0.856	0.953
B–D	0.790	0.674	0.680	0.918
C–D	0.870	0.923	0.916	0.927

Tabela 1.3 pokazuje korelacje pomiędzy poszczególnymi wartościami. Przedstawia ona szereg interesujących zależności:

A*–*B *wysoka korelacja wskazuje, że wykorzystanie wzoru (1.21) jest uzasadnione (w obydwu podejściach **A** i **B** nauczone są wszystkie klasyfikatory potomne Cl^l),*

A*–*C *tu wysoka korelacja pokazuje, że wykorzystanie wzoru (1.21) jest uzasadnione dla aproksymacji $R[HCOC]$ bez rzeczywistego nauczania klasyfikatorów potomnych, jak to się dzieje w **B**; to pokazuje też, że (1.21) może być użyte jako funkcja dopasowania (ang.* fitness*) w wykorzystywanej metodzie szukania macierzy klastrowania,*

A*–*D *wysokie korelacje wskazują na poprawność wykorzystania aproksymacji (1.18) dla aproksymacji działania Cl^l,*

B*–*C *ze względu na małe różnice między **B** i **C** korelacje są wysokie; ta wartość pokazuje jakość estymacji macierzy M^l,*

B*–*D *wysokie wartości wskazują, że prawdziwe macierze M^l uzyskane z nauczania są podobne do macierzy $\tilde{M}^l$ uzyskanych przez estymację, wykorzystując (1.18),*

C*–*D *korelacja między tymi wartościami wskazuje na związek między różnymi sposobami ewaluacji macierzy M^l z wzoru (1.18) i przy wykorzystaniu algorytmu 1.5 – tu wysokie korelacje pokazują, że prosty algorytm 1.5 jest bardzo zbieżny z prostym uśrednieniem.*

*Oczywiście najbardziej interesujące są wartości korelacji **A**–**C**. W tabeli 1.4 podane są doświadczalne wartości tej korelacji dla różnych problemów i dla różnych wielkości sieci. Wyniki pokazują, że korelacje są niezależne od architektury poszczególnych klasyfikatorów bazowych.*

Te wyniki, połączone z doświadczeniami, wskazują, że aproksymacja macierzy pomyłek według (1.18) i aproksymacja błędu (1.21) pozwalają na oszacowanie na podstawie nauczonego już klasyfikatora w korzeniu, jak będą działać klasyfikatory w następnej warstwie. To umożliwi, po zbudowaniu Cl^0 i znalezieniu klastrowania, oszacowanie wartości $R[HCOC]$ po dodaniu kolejnej warstwy.

Ponieważ wiadomo, że działanie całego HCOC będzie zależeć od działania klasyfikatorów w węzłach, można się pokusić o zbudowanie kilku różnych klasyfikatorów w korzeniu, oszacować wartości $R[HCOC]$ wynikające z użycia każdego z nich i wybrać najbardziej obiecujący.

Uwaga 1.11. *Naturalnie wyniki znalezione w eksperymencie powyżej należy przyjmować z ostrożnością, aby nie wysnuć nieprawdziwego wniosku, że wystarczy zbudować klasyfikator w korzeniu, a resztę symulować. Tak zbudowany HCOC nie pozwoliłby na poprawianie wyników. Dodatkowo, jak zauważa wielu autorów, składanie klasyfikatorów ma sens tylko wtedy, gdy są one różne, a w szczególności popełniają błędy w różnych obszarach rozwiązywanego problemu (Kuncheva, 2005; Shipp, Kuncheva, 2002; Giacinto, Roli, 2001). Symulowane klasyfikatory pochodne są naturalnie skorelowane z sobą. Nauczanie potomnych Cl^l jest więc konieczne, jednak pokazana ewaluacja ułatwia wybór architektury.*

Tabela 1.4. Średnie korelacje Spearmana ρ dla porównania **A–C** dla wszystkich sieci (część górna), dla sieci z ustaloną liczbą neuronów ukrytych w sieci Cl^0 w korzeniu (część środkowa) i dla sieci z ustaloną liczbą neuronów ukrytych w klasyfikatorach potomnych Cl^l i różną w Cl^0

	Tumor	Vowel	Audiology	Arrhythmia
Wszystkie	0.843	0.690	0.803	0.868
Cl^0				
2	0.910	0.669	0.804	0.904
3	0.872	0.664	0.781	0.871
4	0.798	0.667	0.796	0.873
5	0.745	0.641	0.795	0.858
6	0.716	0.673	0.744	0.846
Cl^l				
2	0.821	0.794	0.824	0.879
3	0.891	0.861	0.843	0.882
4	0.933	0.886	0.859	0.887

1.3. Wykorzystanie słabych klasyfikatorów

Po raz pierwszy pojęcie *słabego nauczania* pojawiło się w pracach (Valiant, 1984; Kearns, Valiant, 1989, 1994), będących podstawą do tzw. PAC-learning (ang. *Probably Approximately Correct*). W tym modelu nauczanie odbywa się na losowo wybranych przykładach i zbudowana hipoteza musi być poprawna na wszystkich, poza niewielką liczbą, przykładach.

Definicja *słabego* nauczania nie wymaga, aby zbudowany model był w stanie rozpoznawać wszystkie, poza skończoną liczbą, przykłady. Schapire (1990) pokazał równoważność nauczania słabego i nauczania silnego poprzez zdefiniowanie konstrukcji klasyfikatora silnego z wielu słabych. Dało to początek podejściu do nauczania maszynowego poprzez tzw. *boosting*, a więc naprawianie wyników jednego słabego klasyfikatora przez zbudowanie szeregu kolejnych, także słabych, klasyfikatorów. Każdy słaby klasyfikator powinien być prosty w konstrukcji. Okazało się więc, że istnieje możliwość budowy poprawnych i silnych klasyfikatorów z wykorzystaniem bardzo prostych środków.

Pierwsze podejście było tzw. boostingiem przez *filtrowanie*. Później zaproponowany został algorytm AdaBoost, będący złożeniem sekwencji słabych klasyfikatorów i osiągający praktycznie nieograniczoną dokładność (Freund, Schapire, 1997). AdaBoost, oryginalnie zdfiniowany dla problemów dwuklasowych, docze-

kał się rozszerzeń dla problemów wieloklasowych (m.in. Friedman *et al.*, 2000, 2004; Eibl, Pfeiffer, 2005; Bartlett *et al.*, 2006; Bühlmann, Yu, 2006).

Definicja 1.12. Jeśli wartość ryzyka klasyfikatora jest niewiele niższa od wartości ryzyka dla klasyfikatora losowego, to klasyfikator nazywamy *słabym*.

Ta definicja jest oczywista dla liczby klas $K = 2$. Wartość ryzyka, tzn. wartości oczekiwanej funkcji kosztu na poziomie $1/2 - \epsilon$, $\epsilon > 0$, oznacza, że Cl działa niewiele lepiej niż losowo. Klasyfikator Cl rozpoznający $K > 2$ klasy i osiągający błąd rzędu $1/2$ jest w rzeczywistości znacznie silniejszy. Całkiem losowy Cl dla $K > 2$ klas będzie miał błąd rzędu $1 - 1/K$. Dla poradzenia sobie z większą liczbą klas Freund, Schapire (1997) zaproponowali miarę *pseudo-loss*.

Definicja 1.13. (Freund, Schapire, 1997) Funkcja kosztu *pseudo-loss* ma postać

$$\ell(x, C_{tr(x)}, h(x)) = \frac{1}{2}\left(1 - h(x, C_{tr(x)}) + \frac{1}{K-1}\sum_{C \neq C_{tr(x)}} h(x, C)\right), \tag{1.22}$$

gdzie $C_{tr(x)}$ oznacza prawdziwą (*tr* jak *true)* klasę przykładu x, $h(x, C_{tr_{(x)}})$ jest wartością hipotezy, że prawdziwą klasą x jest $C_{tr(x)}$, a $h(x, C)$ jest wartością hipotezy, że klasą jest C.

Ta definicja odpowiada testowaniu par alternatyw, czy hipoteza $h()$ wskazuje na klasę $C_{tr(x)}$ czy na inną (nieprawidłową) C. Dla każdej pary $C_{tr(x)}$ i $C \neq C_{tr(x)}$ zachodzi $h(x, C_{tr(x)}) + h(x, C) = 1$. Jest to wbudowanie do funkcji kosztu schematu wielokrotnego klasyfikatora *jeden-przeciwko-wszystkim* (ang. *one-against-all*). Klasyfikator, dla którego wartość *pseudo-loss* jest niewiele niższa od $1/2$, nazywamy słabym (*weak*).

Funkcja *pseudo-loss* ma wartości z przedziału $[0, 1]$. Jeśli $h(x, C_{tr(x)}) = 1$, to $\ell(x, C_{tr(x)}, h(x)) = 0$. Jeśli wartość $h(x, C_{tr(x)}) = 0$, to $\ell(x, C_{tr(x)}, h(x)) = 1$. Jeśli $h(x, C_{tr(x)}) = 1/K$, to $\ell(x, C_{tr(x)}, h(x)) = 0.5$. Wartość tak zdefiniowanej funkcji kosztu będzie tym niższa, im aktywacja dla prawdziwej klasy jest wyższa od *średniej* aktywacji na pozostałych pozycjach. Jeśli jest równa średniej, to wartość $\ell(x, C_{tr(x)}, Cl(x)) = 1/2$. Użycie tej funkcji kosztu leży u podstaw działania algorytmu AdaBoost.M2 (Freund, Schapire, 1997). Tak więc jeśli wartość oczekiwana aktywacji dla prawidłowej klasy będzie bliska *średniej* aktywacji dla innych klas, to wartość funkcji *pseudo-loss* będzie bliska 0.5.

Definicja 1.14. Klasyfikator jest słaby, jeśli wartość oczekiwana funkcji *pseudo-loss* jest niższa od $1/2$.

Ta definicja nie bierze pod uwagę, że po obliczeniu aktywacji, zgodnie z definicją klasyfikatora 1.1, konieczny jest wybór końcowej klasy za pomocą jakiejś reguły (funkcja selekcji S w definicji 1.1). Zwykle będzie to reguła typu arg max wyboru klasy o najwyższej aktywacji. Definicja uwzględnia jedynie *średnie* poziomy aktywacji, a nie jak często każda z klas była poprawnie wybierana. Okazuje

się, że zbyt mały margines aktywacji ponad wartość $1/K$ lub ponad średnią aktywację na pozostałych pozycjach nie jest wystarczający, aby dana klasa została wybrana przez S z prawdopodobieństwem większym od $1/K$.

Przykład 1.15. *Niech dany będzie trzyklasowy problem. X_k jest podzbiorem tych przykładów, które należą do klasy C_k. Niech wartość oczekiwana aktywacji dla $x \in X_1$ wynosi*

$$E[Cl(x) \mid x \in X_1] = [0.35, 0.39, 0.29].$$

W oczywisty sposób aktywacje na pierwszej pozycji są wyższe od $1/K = 1/3$ i od średniej aktywacji na pozostałych pozycjach $(0.39 + 0.29)/2 = 0.33$. Nie oznacza to jednak, że klasa C_1 zostanie wybrana. Przewaga tej wartości ponad wartość średnią jest zbyt mała.

Tak zdefiniowany klasyfikator będzie można nazwać słabym zgodnie z definicją 1.14, ponieważ dla x z klasy C_1 mamy $E[\ell(x, C_1, Cl|x \in C_1)] = (1 - 0.35 + 0.33)/2 = 0.49$ (podobnie dla pozostałych klas), jednak prawdopodobieństwo, że częściej niż niepoprawna będzie wybrana poprawna klasa, jest mniejsze od $1/2$.

Okazuje się, co będzie pokazane niżej, że prawdopodobieństwo wybrania poprawnej klasy wynosi w tym przykładzie zaledwie 0.27869, czyli będzie mniejsze od $1/K = 1/3$.

Definicja 1.16. Niech Cl będzie K-klasowym klasyfikatorem. Niech Cl wykorzystuje do selekcji klasy podejście arg max (wybiera klasę o najwyższej aktywacji).

Cl nazywamy *słabym* klasyfikatorem, jeśli prawdopodobieństwo, że aktywacja dla prawdziwej klasy jest *wyższa* od którejkolwiek innej aktywacji z prawdopodobieństwem wyższym od $1/K$:

$$Pr(Cl_{tr(x)}(x) > Cl_j(x)\ \forall\, j \neq tr(x)) > 1/K. \tag{1.23}$$

Definicja 1.17. K-klasowy klasyfikator Cl nazywamy słabym, jeżeli wartość oczekiwana aktywacji na pozycji poprawnej klasy jest większa od $\alpha(K)$, gdzie

$$\alpha : \mathbb{N} \longrightarrow (0, 1] \tag{1.24}$$

zwraca minimalną wartość aktywacji taką, że wartość po lewej stronie wyrażenia (1.23) jest równa $1/K$.

Przy założeniu, że średnie wartości aktywacji na wszystkich pozycjach poza prawidłową są równe, można udowodnić twierdzenie o minimalnej wartości $\alpha(K)$. To założenie nie będzie zwykle spełnione, niezależnie jednak od tego proponowane tu podejście podaje lepsze oszacowanie minimalnej aktywacji.

Twierdzenie 1.18. *Niech Cl będzie K-klasowym klasyfikatorem, $K > 2$. Niech wartości oczekiwane aktywacji na pozycjach różnych od poprawnej będą równe.*

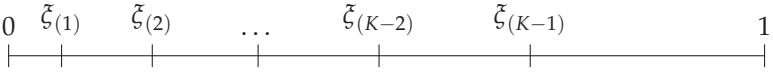
Cl jest słaby wtedy i tylko wtedy, gdy $\sum_{i=1}^{K} Pr(C_i)E[Cl_i(x)|tr(x) = i] > \alpha(K)$, gdzie

$$\alpha(K) = \min_{\alpha} \left\{ \alpha : \sum_{i=0}^{K-2} (-1)^i \binom{K-1}{i} \left(1 - \frac{i\alpha}{1-\alpha}\right)_+^{K-2} > \frac{1}{K} \right\}, \tag{1.25}$$

gdzie p_i jest prawdopodobieństwem a priori *klasy C_i, $(x)_+ = \max(0, x)$. Niech $\alpha(1) = 1$ z definicji.*

Dowód. (Podolak, Roman, 2009) Niech Cl będzie K-klasowym klasyfikatorem dającym $[y_1, \ldots, y_K]$ jako wektor prawdopodobieństw klas $P(C|x)$. Z założenia wektor długości przedziałów $[y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_K]$ (dla wygody zapisu niech $t \neq K$) ma rozkład równomierny. $\sum_{k \neq t} y_k = 1 - y_t$. Niech $\alpha = y_t$, gdzie t jest indeksem prawdziwej klasy.

Niech $y_k, k \neq t$, będą długościami przedziałów związanymi z podziałem odcinka $I = [0, 1 - y_t]$. Niech $\xi_1, \ldots, \xi_{K-2}$ będą zmiennymi losowymi odpowiadającymi końcom przedziałów związanych ze zmiennymi y_i dzielących przedział I i niech $\xi_0 = 0$ i $\xi_{K-1} = 1 - y_t$. Możemy zastosować permutację $\xi_{(1)}, \ldots, \xi_{(K-2)}$ zmiennych $\xi_1, \ldots, \xi_{K-2}$ taką, że $\xi_{(1)} \leq \cdots \leq \xi_{(K-2)}$. Ustalmy $\xi_{(0)} = 0$ oraz $\xi_{(K-1)} = 1 - y_t$. Wartości $\xi_{(i)}$ odpowiadają punktom podziału $[0, 1 - y_t]$ tak, że odcinki odpowiadają kolejno aktywacjom w kolejności rosnącej.



Niech $\{\eta_{(i)} = \xi_{(i)} - x_{(i-1)}\}_{i=1}^{K-1}$ będzie rodziną statystyk $\eta = (\eta_{(1)}, \ldots, \eta_{(K-1)})$. Jeśli ξ_i są zmiennymi losowymi na $[0, 1]$, to można każdy przedział o długości l przeskalować do $[0, 1 - y_t]$, wykorzystując funkcję skalującą $s(l) = l/(1 - y_t)$.

Rozkład η jest rozkładem Dirichleta (patrz dodatek A.12)

$$(\eta_{(1)}, \ldots, \eta_{(K-2)}) \simeq Dir(1, \ldots, 1; 1), \tag{1.26}$$

gdzie $Dir(a_1, \ldots, a_{K-2}; a_{K-1})$ jest rozkładem Dirichleta parametryzowanym przez $a_1, \ldots, a_{K-1}$. W naszym przypadku, gdy rozkład aktywacji dla wszystkich klas poza prawdziwą klasą C_t jest z założenia równomierny, mamy wszystkie $a_i = 1$. Gęstość prawdopodobieństwa wektora prawdopodobieństw $(\eta_{(1)}, \ldots, \eta_{(K-2)})$ ma wartość $f(x_1, \ldots, x_{K-2}; 1, \ldots, 1) = (K-2)!$.

Warunek, aby klasyfikator Cl poprawnie wybrał dla przykładu $x \in X_t$ klasę t jako prawdziwą (czyli dla x należącego do klasy C_t), jest równoważny warunkowi

$$Pr\{\arg\max_i \; y_i^{Cl}(x) = t \mid t\} = Pr\{y_t > y_i \forall\, i \neq t \mid t\}. \tag{1.27}$$

Interesuje nas najmniejsza wartość y_t (czyli $\alpha(K)$) taka, że to prawdopodobieństwo będzie większe od $1/K$, tak jak to jest określone w definicji 1.17. Szukając α dla przedziału $[0, 1 - y_t]$, wtedy dla przedziału $[0, 1]$ szukamy wartości $\alpha/(1 - \alpha)$, używając funkcji skalującej $s(l)$ dla $l = \alpha$.

Niech α będzie ustalone. Wtedy

$$Pr\left\{ \frac{\alpha}{(1-\alpha)} > y_i \;\forall\, i \neq t \mid t \right\} = \int_A (K-2)!\, dx_1 dx_2 \ldots dx_{t-1} dx_{t+1} \ldots dx_{K-1},$$

gdzie

$$A = \left\{ (x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_{K-1}) : \forall\, i \neq t \; x_i < \frac{\alpha}{1-\alpha} \wedge \sum_{i \neq t} x_i = 1 - \frac{\alpha}{1-\alpha} \right\}.$$

Rozkład brzegowy $\eta_{(i)}$ wektora $\eta = (\eta_{(1)}, \ldots, \eta_{(K-1)})$ ma postać rozkładu Beta $Beta(1, \sum_j \alpha_j - \alpha_i) = Beta(1, K-2)$ (Ferguson, 1973). Funkcja gęstości prawdopodobieństwa rozkładu Beta ma postać

$$Beta(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}. \tag{1.28}$$

Podstawiając $a = 1$ i $b = K-2$ dla każdego $i = 1, \ldots, K-1$, otrzymujemy

$$\begin{aligned} Pr\{\eta_{(i)} < x\} &= \int_0^x \frac{\Gamma(1+K-2)}{\Gamma(1)\Gamma(K-2)} u^{1-1}(1-u)^{K-2-1}du \\ &= \int_0^x \frac{\Gamma(K-1)}{\Gamma(1)\Gamma(K-2)} u^{1-1}(1-u)^{K-2-1}du \\ &= \int_0^x \frac{(K-2)\Gamma(K-2)}{\Gamma(K-2)}(1-u)^{K-3}du \\ &= \int_0^x (K-2)(1-u)^{K-3}du = 1-(1-x)^{K-2}. \end{aligned} \tag{1.29}$$

Stąd $Pr\{\eta_{(i)} > x\} = (1-x)^{K-2}$.

Można zauważyć, że

$$Pr\{\eta_{(1)} > x_1, \ldots, \eta_{(K-1)} > x_{K-1}\} = (1 - x_1 - x_2 - \cdots - x_{K-1})_+^{K-2}, \tag{1.30}$$

gdzie $(a)_+ = \max(0, a)$.

Wykorzystując zasadę włączeń–wyłączeń, otrzymujemy

$$Pr\left\{\eta_{(1)} < \frac{\alpha}{1-\alpha}, \ldots, \eta_{(K-1)} < \frac{\alpha}{1-\alpha}\right\} = \sum_{i=0}^{K-2}(-1)^i \binom{K-1}{i}\left(1 - \frac{i\alpha}{1-\alpha}\right)_+^{K-2}. \tag{1.31}$$

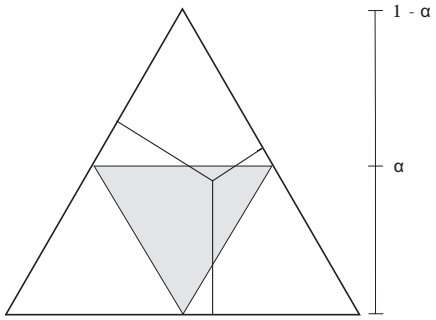
Jeśli ta wartość prawdopodobieństwa jest większa od $1/K$, to mówimy, że klasyfikator jest słaby. □

Niektóre wartości $\alpha(K)$ dla małych K podane są w tabeli 1.5.

Przykład 1.19. *Ideę definicji słabego klasyfikatora pokazuje rysunek 1.8. Problem znalezienia $\alpha(K=4)$ można przedstawić jako geometryczny. Niech będzie dany trójkąt równoboczny o wysokości $1-\alpha$ z wrysowanym w środku mniejszym trójkątem o wysokości α. Definicji 1.17 odpowiada rozwiązanie następującego problemu: jaka jest minimalna wartość α taka, żeby punkt wylosowany z trójkąta z prawdopodobieństwem co najmniej $1/K = 1/4$ był wylosowany z wewnętrznego trójkąta? To odpowiada rozwiązaniu*

$$\frac{\alpha^2/2}{(1-\alpha)^2/2} = \frac{1}{4},$$

co daje $\alpha(4) = 1/3$.



Rysunek 1.8. Wartość $\alpha(K)$ dla $K = 4$

Tabela 1.5. Kilka początkowych wartości $\alpha(K)$

K	$\alpha(K)$	K	$\alpha(K)$	K	$\alpha(K)$	K	$\alpha(K)$	K	$\alpha(K)$	K	$\alpha(K)$
1	1.000	5	0.289	9	0.194	13	0.150	17	0.124	21	0.106
2	0.500	6	0.256	10	0.180	14	0.142	18	0.119	22	0.103
3	0.400	7	0.230	11	0.169	15	0.135	19	0.114	23	0.099
4	0.333	8	0.210	12	0.159	16	0.129	20	0.110	24	0.096

Obserwacja 1.20. *Na rysunku 1.9 występują różne zależności $\alpha(K)$. Wartości $\alpha(K)$ oraz $1/K$ są równe dla $K = 1$ i $K = 2$, dla wyższych $\alpha(K) > 1/K$. Pokazuje to zwiększone wymaganie co do słabości klasyfikatora wynikające z twierdzenia 1.18. Zgodnie z definicją 1.1 klasyfikatora, po obliczeniu wartości prawdopodobieństwa przynależności do klas następuje etap wyboru, a wymagania co do średniej aktywacji okazują się wyższe. Największa bezwzględna różnica $\alpha(K) - 1/K$ jest dla $K = 5$ (różnica prawie* 0.09*).*

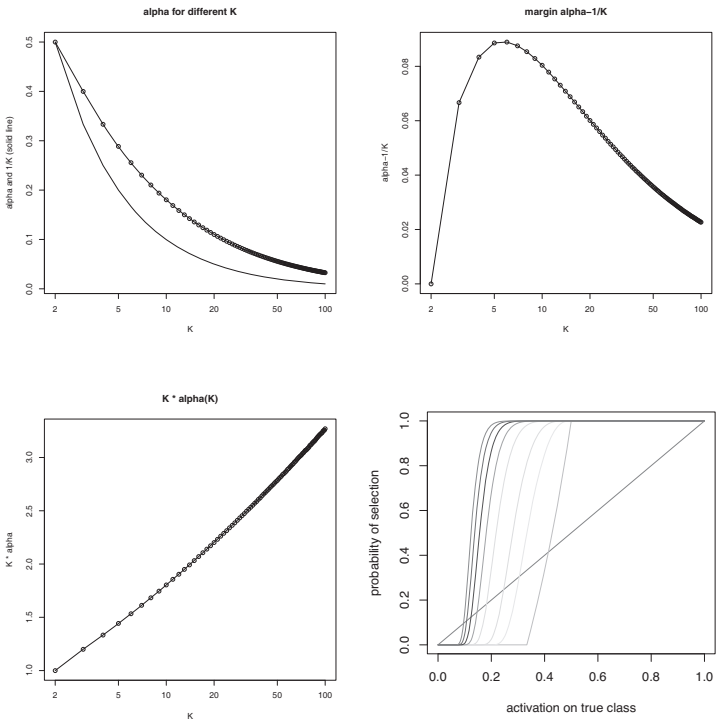
Jednocześnie warto zauważyć, że $K * 1/K$ jest wartością stałą, $K * \alpha(K)$ natomiast jest ściśle rosnąca, co przedstawia coraz bardziej zwiększone wymagania narzucone przez nową definicję wraz ze wzrostem liczby klas.

Ostatni wykres na rysunku 1.9 pokazuje zależność prawdopodobieństwa wyboru danej klasy od wartości aktywacji w zależności od liczby klas K. Widać wyraźnie wyższe wymagania narzucone przez wprowadzony formalizm $\alpha(K)$.

Obliczanie wartości $\alpha(K)$ dla większych K wprost z twierdzenia jest pracochłonne. Można jednak dobrze przybliżyć wartość $\alpha(K)$, korzystając z modelu ax^b. Rozwiązanie ma postać

$$\alpha(K) \simeq 0.83723K^{-0.68566}. \tag{1.32}$$

Własność słabości klasyfikatora bierze pod uwagę wartość oczekiwaną aktywacji na poprawnych pozycjach dla wszystkich klas. Czasem będzie potrzebna modyfikacja tej definicji.



Rysunek 1.9. Wartości $\alpha(K)$ dla różnych K. Od góry do dołu: $\alpha(K)$ (linia kropkowana) względem $1/K$ (linia ciągła), margines $\alpha(K) - 1/K$, $K * \alpha(K)$. Wartości liczby klas w skali logarytmicznej. Ostatni rysunek przedstawia prawdopodobieństwa wyboru klasy w zależności od aktywacji dla $K = 3, 5, 7, 11, 13, 15, 19, 23, 27$. Dla porównania liniowa struktura *pseudo-loss*

Definicja 1.21. Klasyfikator *Cl* nazywamy słabym w sposób silny, jeżeli jest słaby dla każdej klasy z osobna.

Ta definicja wzmacnia poprzednią w tym sensie, że w definicji zasadniczej klasyfikator może być nawet bardziej losowy dla niektórych klas, a mimo to wartość oczekiwana aktywacji spełni wymienione warunki. Będzie to miało miejsce wtedy, gdy w problemie istnieją klasy dominujące, reprezentowane przez znacznie większą liczbę przykładów. *Cl* może wtedy w prosty sposób nauczyć się tak, aby oczekiwana wartość aktywacji spełniała warunki definicji słabego klasyfikatora, a jednocześnie mniej się przystosowywać do rzeczywistego problemu.

Przykład 1.22. *Klasyfikatory typu zero-rule budowane są w następujący sposób: algorytm zlicza liczbę reprezentantów każdej z klas i wyszukuje klasę C_s, która występuje najczęściej. Następnie dla każdego przykładu zwraca jako odpowiedź właśnie C_s.*

Niech zbiór uczący ma N przykładów, a klasa C_s występuje w nim N_s razy. Jeśli przybliżymy wartość oczekiwaną przez prostą średnią, to aktywacja na poprawnej pozycji będzie wynosić

$$(N_s \cdot 1 + (N - N_s) \cdot 0)/N = N_s/N$$

i wystarczy spełnienie $N_s/N > 1 - \alpha(K)$ dla K możliwych klas, aby klasyfikator był słaby w sensie podanej definicji (dla mniejszej liczby klas może się to zdarzyć często). Ponieważ w tym specyficznym przykładzie aktywacje są równe albo 1, *albo* 0, *wystarczy, aby $N_s/N > 1 - 1/K$, co jest słabszym wymaganiem.*

Zbudowanie modelu HCOC opartego na klasyfikatorach takich jak zero-rule w węzłach będzie się mijać z celem. Dla każdego przykładu, w danym węźle, odpowiedź będzie identyczna, a w związku z tym zupełnie niezwiązana z przykładem klasyfikowanym. Sama słabość klasyfikatora bazowego nie będzie więc wystarczająca do zbudowania zadowalającego modelu HCOC.

Jeśli w węzłach HCOC wykorzystane zostaną klasyfikatory słabe w sposób silny tak jak w ostatniej definicji, pozwoli to na lepsze oszacowanie wartości ryzyka $R[HCOC]$ (patrz twierdzenie 2.33 w rozdziale 2.7.1).

1.4. Podsumowanie i uwagi

W tym rozdziale pokazane zostało, w jaki sposób możliwy jest podział zadania na mniejsze podzadania przy jednoczesnej aproksymacji oczekiwanej wartości funkcji kosztu, którą osiągnąłby HCOC, gdyby taki podział zastosowano. Pozwala to na nauczenie kilku klasyfikatorów w korzeniu, aproksymację wartości $R[HCOC]$ i wybór tego Cl^0 w korzeniu, który daje najlepsze szanse na osiągnięcie optymalnego wyniku końcowego.

Jednocześnie taka miara może służyć jako funkcja dopasowania w tworzącym taki podział algorytmie klastrowania. W następnej części w rozdziale 2.6 podane są szczegółowe propozycje tych algorytmów.

Na końcu tego rozdziału przedstawiona została nowa definicja słabego klasyfikatora, różna od obecnych w literaturze. Ta definicja ma bezpośredni wpływ na budowę HCOC, ponieważ będzie miarą mówiącą, w jaki sposób musi być nauczony każdy klasyfikator bazowy. Pozwoli to na ustrzeżenie się przed sytuacjami, w których klasyfikator w jednym z węzłów uniemożliwi dobry wynik końcowy HCOC.

Warunek, aby klasyfikatory bazowe w węzłach były słabe, jest podstawowy dla modelu HCOC. HCOC jest modelem łączenia wyników pośrednio zależnym od danych: waga każdego klasyfikatora rozwiązującego podproblem nie jest stała, lecz jest zależna od aktualnie klasyfikowanego przykładu (patrz dodatek A.6). Dzięki temu, że klasyfikator jest słaby, wagi klasyfikatorów zawierających w swoim podproblemie prawidłową klasę rozpatrywanego przykładu będą miały wagi istotnie większe od tych dla podproblemów, które nie zawierają jej. Tak więc dzięki temu założeniu szum pochodzący od „nieprawidłowych“ podproblemów

(tj. niezawierających właściwej klasy) będzie minimalizowany przez mniejsze wagi. To pokaże, że klasyfikator *uogólniony* – wskazujący prawidłowy podproblem – przy zachowaniu warunku słabości klasyfikatorów bazowych będzie „silny", a więc będzie osiągał niski błąd. W tej konstrukcji zostanie wprost użyta definicja wartości $\alpha(K)$. W rozdziale 2.3.4 jest to szczegółowo omówione.

W następnej części podana będzie pełna definicja Hierarchicznego Klasyfikatora, jego elementy i algorytmy pozwalające na ich budowę. Pokazane zostanie także, że wraz z dodawaniem kolejnych węzłów wartość funkcji ryzyka maleje.

Rozdział 2

Klasyfikator Hierarchiczny HCOC

Jak było opisane w rozdziale 1, w problemie klasyfikacji wykorzystywane są dwa podstawowe podejścia do podziału zadania. Pierwszym jest podział przestrzeni wejściowej $\mathcal{X}$, rozwiązanie zadanego problemu dla każdej części osobno, a następnie złożenie wyników dla uzyskania końcowej odpowiedzi. Podział $\mathcal{X}$ może być *twardy* (ang. *crisp*), poprzez wyznaczenie granic w $\mathcal{X}$, np. przez klastrowanie algorytmem *K*-średnich. Dla każdej wyznaczonej podprzestrzeni $\mathcal{X}_i$ utworzony zostanie klasyfikator Cl^i. Dla predykcji wyniki klasyfikacji będą złożone ustalonym algorytmem. Podział przestrzeni wejściowej może być także *miękki* (ang. *soft*), gdy istnieją $i, j, i \neq j$ takie, że $\mathcal{X}_i \cap \mathcal{X}_j \neq \emptyset$, gdzie $\mathcal{X}_i$ oznacza pewien podzbiór całego zbioru atrybutów.

Drugim możliwym podejściem będzie podział przestrzeni *wyjściowej* klas $\mathcal{C} = \{C_1, \ldots, C_K\}$ na podproblemy $\mathcal{C}_l$, $l = 1, \ldots, L$. Ten podział może być twardy albo też miękki. Takie rozwiązanie przyjmuje właśnie, opisywany w tej pracy, HCOC. Idea hierarchicznego podziału $\mathcal{C}$ na kolejne nakładające się podproblemy wynika z obserwacji wyników klasyfikacji dla problemów, w których liczność zbioru klas $\mathcal{C}$ jest stosunkowo duża.

2.1. Podział przestrzeni wyjściowej klas

W modelach składających się z wielu prostszych klasyfikatorów celem jest osiągnięcie *zysku* polegającego na tym, że wartość ryzyka dla całego klasyfikatora będzie niższa niż dla jego składowych. W przypadku HCOC, który ma strukturę drzewiastą, będziemy żądać, aby $R[HCOC] < R[Cl^0]$, gdzie Cl^0 jest klasyfikatorem w korzeniu.

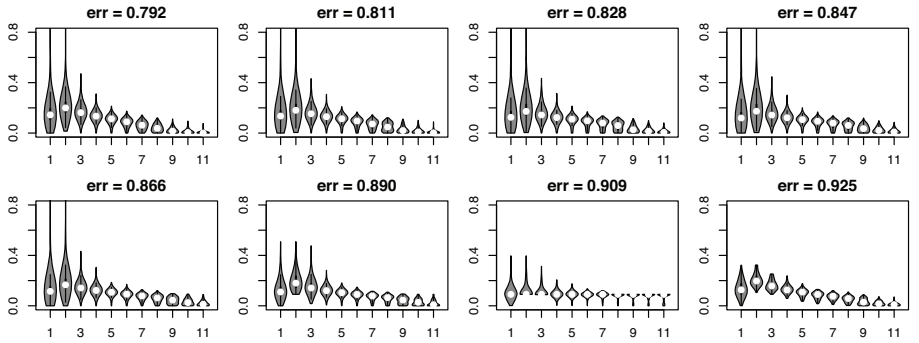
W rozdziale 1.3 omówiony był klasyfikator słaby, który ma stosunkowo dużą wartość ryzyka, jednak niższą od wartości ryzyka dla klasyfikatora losowe-

go. Słaby klasyfikator ma pewne szczególne własności, które uzasadniają podział przestrzeni klas.

Obserwacja 2.1. *Niech Cl będzie klasyfikatorem. Dla wektora atrybutów x wektor*

$$Cl(x) = (Cl_1(x), \ldots, Cl_K(x)) \tag{2.1}$$

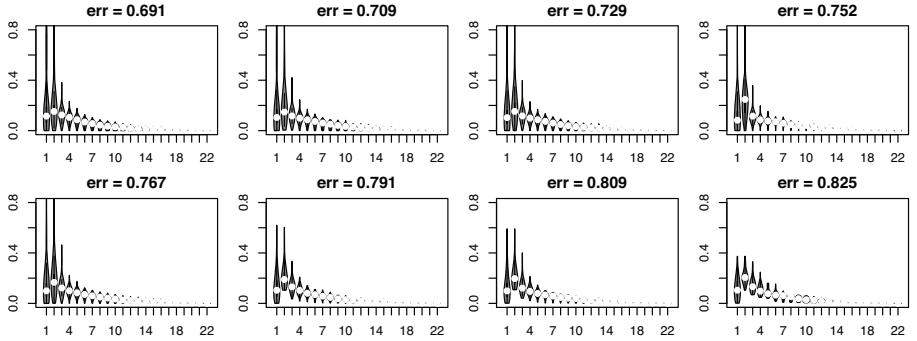
zgodnie z definicją 1.1 jest traktowany jako wektor prawdopodobieństw klas a posteriori $P(C_i|x)$. *Przy wykorzystaniu reguły* arg max *jako końcową odpowiedź wybierana jest klasa, dla której wartość $Cl(x)$ jest najwyższa (Hastie* et al.*, 2001). Takie jest zadanie funkcji $S()$ w definicji 1.1. Jaka jest szansa, że dla klasyfikatora słabego będzie to klasa poprawna? Dla problemów dwuklasowych ten wybór jest oczywisty, jednak jeśli liczba klas $K > 2$, już tak nie jest.*



Rysunek 2.1. Statystyka różnych słabych klasyfikatorów dla problemu vowel (11 klas wyjściowych) (Newman *et al.*, 1998). U góry wielkość średniego błędu. Dla każdego klasyfikatora pierwszy od lewej wykres gęstości odpowiada średniej aktywacji dla poprawnej odpowiedzi, a kolejne odpowiadają średnim aktywacjom posortowanym malejąco według wartości (osobno dla każdej klasy). Wyniki pochodzą z uśrednienia wielu modeli zbudowanych dla tego samego problemu. Losowy klasyfikator dla tego problemu ma błąd $1 - 1/11 \simeq 0.909$. Słaby klasyfikator w sensie twierdzenia 1.18 ma błąd mniejszy od $1 - \alpha(11) \simeq 0.831$

Na rysunkach 2.1 i 2.2 pokazane są statystyki różnych słabych klasyfikatorów dla problemów vowel i primary-tumour. Istotne jest, że są to klasyfikatory o błędzie w okolicach wartości granicznych dla klasyfikatorów słabych, wynikające z definicji (patrz rozdział 1.3). Można zauważyć, że najwyższą aktywację otrzymuje zwykle pierwsza bądź druga nieprawidłowa klasa, a aktywacja dla prawidłowej klasy jest zwykle na drugim lub trzecim miejscu.

Jeśli aktywacja dla klasy prawidłowej nie jest zwykle tą najwyższą, ale drugą bądź trzecią, to istnieje duże prawdopodobieństwo, że wybierając grupę klas w następujący sposób:



Rysunek 2.2. Statystyka różnych słabych klasyfikatorów dla problemu primary tumor (21 klas wyjściowych). Rysunki dla poszczególnych sieci zorganizowane tak jak na rysunku 2.1. Całkowicie losowy klasyfikator dla tego problemu ma błąd $1 - 1/21 \simeq 0.952$. Słaby klasyfikator ma błąd mniejszy od $1 - \alpha(21) \simeq 0.894$

- dana klasa o średniej aktywacji $\overline{Cl_i}$,
- dwie lub trzy inne klasy o najwyższych średnich aktywacjach, gdy prawdziwa jest klasa C_i,

wyselekcjonujemy grupę, *w której znajdzie się prawdziwa* klasa. Oczywiście taki naiwny sposób tworzenia podproblemów nie jest wystarczający.

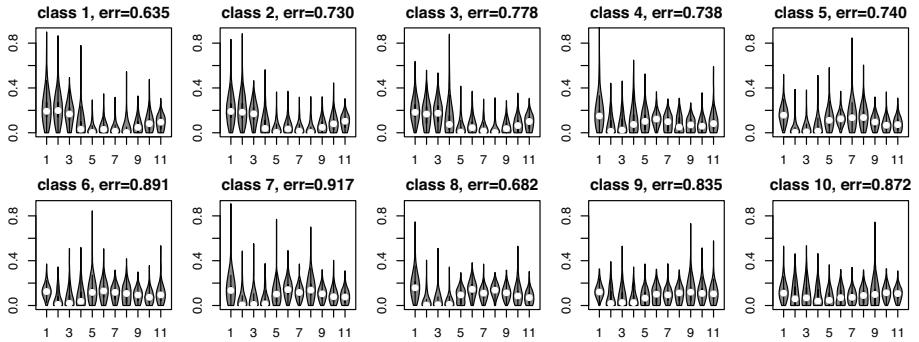
Hipoteza 2.2. *Dla słabego klasyfikatora Cl^i o wysokim błędzie średni wektor aktywacji ma naturalne odchylenie w kierunku odpowiedzi dla klas, które są bardziej liczne w zbiorze przykładów uczących $\mathcal{D}$. Średni wektor odpowiedzi $[\overline{P}(C_1|x, Cl), \ldots, \overline{P}(C_K|x, Cl)]$ klasyfikacji przykładów z danej klasy C_k jest odchylony w kierunku średnich wektorów odpowiedzi dla klas, które mają wspólne cechy z C_k.*

Hipoteza 2.3. *Najbardziej istotna jest odległość (w sensie przyjętej miary, np. euklidesowej) między wektorami odpowiedzi, które są wektorami prawdopodobieństw przynależności do klas. Jeśli odległości między odpowiedziami dla przykładów z różnych prawdziwych klas są niewielkie, to klasy są do siebie prawdopodobnie podobne, a stąd trudniejsze jest ich rozróżnienie. Jeśli odległości między wektorami aktywacji są większe, to klasy są prawdopodobnie mniej do siebie podobne i prostsze do rozróżnienia.*

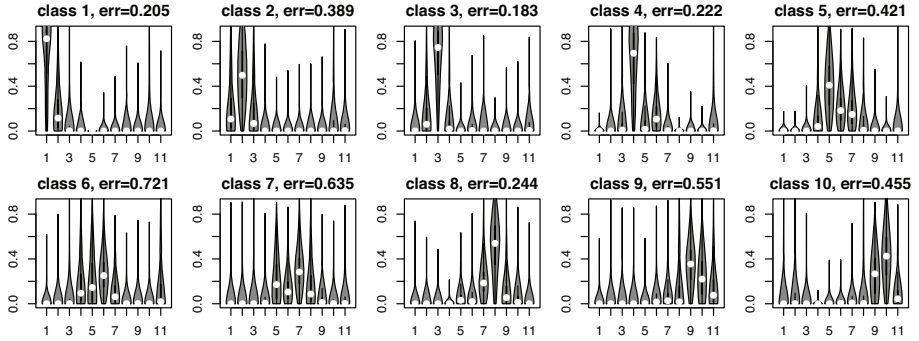
W pierwszym wymienionym w hipotezie 2.3 przypadku klasy odpowiadające tym przykładom powinny być przypisane do jednej grupy i rozróżniane na następnym poziomie złożonego klasyfikatora. W drugim przypadku powinny być przypisane do różnych grup. Odpowiada to zadaniu *klastrowania*. Klastry będą tworzone z często mylonych klas. Ta charakterystyka ma wyraźny związek ze średnim błędem klasyfikatora. Im klasyfikator jest słabszy, tym znajduje się „niżej" na liście rankingowej średnich aktywacji dla prawdziwej klasy. Odchylenie

odpowiedzi ma związek z problemem *bias–variance* (np. Hastie *et al.*, 2001). Klastrowanie będzie odpowiedzialne za podział na grupy (klastry) według średnich tak, aby w klasyfikatorach rozwiązujących tam problem znaleźć rozwiązanie dla minimalizacji wariancji.

Obserwacja 2.4. *Czy jednak aktywacje dla danych klas, w relacji do innych, są podobne dla różnych sieci, które osiągają różny błąd? Będzie to odpowiadało profilowi odpowiedzi dla różnych klas i jego zachowaniu przy zmieniającym się błędzie.*



Rysunek 2.3. Średnie aktywacje sieci nauczonych dla problemu vowel. Wykresy gęstości pokazują średnie aktywacje obliczone z 200 sieci, dające razem błąd rzędu 0.792 na całym zbiorze uczącym. Indeksy u dołu każdego wykresu są indeksami klas problemu. Dla prostoty rysunku wyświetlone jest tylko 10 z wszystkich 11 klas problemu



Rysunek 2.4. Średnie aktywacje sieci nauczonych dla problemu vowel uśrednione z 200 sieci i dające średni błąd 0.391

Rysunki 2.3 i 2.4 pokazują wyniki aktywacji uśrednione z ponad 200 sieci każdy. Łącznie średni błąd sieci na rysunku 2.3 wynosi ok. 0.792 *dla wszystkich klas oraz* 0.391 *dla*

rysunku 2.4. Klasy nie są w jakikolwiek sposób posortowane. Porównując te rysunki, można zauważyć podobny profil odpowiedzi, który ujawnia się przy nauczaniu klasyfikatorów, wykorzystując ten sam zbiór uczący.

Wzorce zachowań klasyfikatorów są wyraźniejsze, gdy błąd jest niski; to naturalne. Ale jeśli profil pozostaje ten sam, to można stosować ten sam wybór, nawet jeśli błąd jest wysoki. Można więc wybrać klaster przez wybór klas o najwyższej aktywacji dla przykładów z ustalonej prawdziwej klasy. W trakcie nauczania klasy wykazujące duże podobieństwo profilu odpowiedzi przy przedstawieniu sieci różnych wejściowych wzorców atrybutów x będą grupowane w klastry $Q \in \mathcal{Q}$.

2.1.1. Klastry i ich nakładanie się

Tworzenie klastrów leży u podstaw modelu HCOC. Niech klasyfikator Cl^0 *często* myli klasy C_j oraz C_k.

Definicja 2.5. Mówimy, że klasyfikator Cl myli klasy C_j i C_k, jeśli aktywacje $Cl(x)$, dla x należących do klasy C_j, mają często najwyższą wartość na pozycji odpowiadającej innej klasie C_k, w związku z czym podejście arg max często będzie je niepoprawnie rozpoznawać.

Mylenie jest związane z wysoką wartością m_{jk} w macierzy pomyłek (oraz, ewentualnie, m_{kj}, jeśli przykłady z prawdziwej klasy C_k będą rozpoznawane jako należące do C_j). Zwykle towarzyszyć temu będą podobne aktywacje na innych pozycjach wektora $Cl(x)$. Algorytm nauczania tworzy więc *klaster* Q taki, że $C_j, C_k \in Q$, który odpowiada podproblemowi złożonemu z tych klas. W trakcie klasyfikacji przykładu przez klasyfikator w korzeniu HCOC (lub jego poddrzewa) wektor x zostanie przekazany do najbardziej prawdopodobnego podproblemu (klastra) Q dla dokładnego rozpoznania i poprawienia ewentualnego błędu.

Jeśli Cl myli C_j oraz C_k, jednocześnie myląc C_k z C_m, to naturalną reakcją algorytmu nauczania byłoby utworzenie klastra $Q \supset \{C_j, C_k, C_m\}$. Jeśli jednak klasy C_j i C_m *nie* są przez Cl mylone, wtedy algorytm powinien utworzyć dwa *różne* klastry: Q_p zawierający (między innymi) klasy C_j i C_k oraz Q_q zawierający C_k i C_m. Mówimy, że te klastry *nakładają* się, stąd pełna nazwa modelu: Hierarchiczny Klasyfikator z Nakładającymi się Grupami Klas (ang. *Hierarchical Classifier with Overlapping Clusters*, HCOC). Dzięki takiemu rozwiązaniu przy podziale na podproblemy wykorzystane jest maksimum informacji z obserwacji działania klasyfikatora Cl w korzeniu aktualnego poddrzewa HCOC.

Uwaga 2.6. *Jeśli chodzi o łączenie w klastrach przykładów najbardziej mylonych, to interesująca jest tu analogia z metodologią drzew decyzyjnych (m.in. Kohavi, Quinlan, 2002; Quinlan, 1993; Breiman, 2001; Quinlan, 1996; Sethi, Sarvarayudu, 1982). W węzłach drzewa decyzyjnego umieszczane są testy atrybutów. Celem algorytmu budowy drzewa jest znalezienie takich sekwencji testów, aby otrzymać możliwie najpłytsze drzewo pozwalające jak najszybciej określić etykietę przykładu.*

W danym kroku rekurencyjnej procedury budowy drzewa algorytm wybiera ten test, który najlepiej rozdziela przykłady uczące na dwie grupy. Odpowiada to więc, podobnie jak w HCOC, jak najprostszemu rozdzieleniu przykładów, pozostawiając trudniejsze problemy do rozwiązania w podproblemie tworzonym przez przykłady z jednej z grup. Jest to równoznaczne z wyszukiwaniem, na danym poziomie, grup klas najbardziej z sobą mylonych, najtrudniejszych do rozróżnienia.

Jednak w danym kroku algorytm budowy drzewa decyzyjnego zakłada niezależność atrybutów i rozpatruje osobne testy dla każdego z atrybutów, aby znaleźć optymalny podział. HCOC rozpatruje jednocześnie aktywacje aktualnego korzenia w celu utworzenia nakładających się podproblemów. HCOC operuje w przestrzeni klas wyjściowych.

2.2. Definicja HCOC

Model klasyfikatora HCOC jest drzewem klasyfikatorów Cl^i takich, że klasyfikator w korzeniu Cl^0 rozwiązuje cały problem Q^0, a klasyfikatory Cl^i, w węzłach niżej, rozwiązują podproblemy.

Definicja 2.7. Niech $\mathcal{D} \subset X \times \mathcal{C}$ będzie zbiorem N przykładów uczących problemy klasyfikacji, a $\mathcal{C} = \{C_i | i = 1, \ldots, K\}_{i=1}^{K}$ zbiorem K możliwych klasyfikacji.

Klasyfikator Hierarchiczny HCOC z nakładającymi się klastrami klas jest trójką

$$HCOC = (V,\ V^0,\ child), \tag{2.2}$$

gdzie

$$\begin{aligned}
child &: V \to 2^V, \\
V &= \{V^i = (Cl^i, F^i)\}, \\
V^0 &\in V, \\
Cl^i &: X \to [0,1]^K, \\
F^i &= \begin{cases} K \times L^i \text{ macierz}(L^i = |child(V^i)|) & \text{jeśli } |child(V^i)| > 0 \\ \emptyset & \text{w przeciwnym przypadku,} \end{cases}
\end{aligned}$$

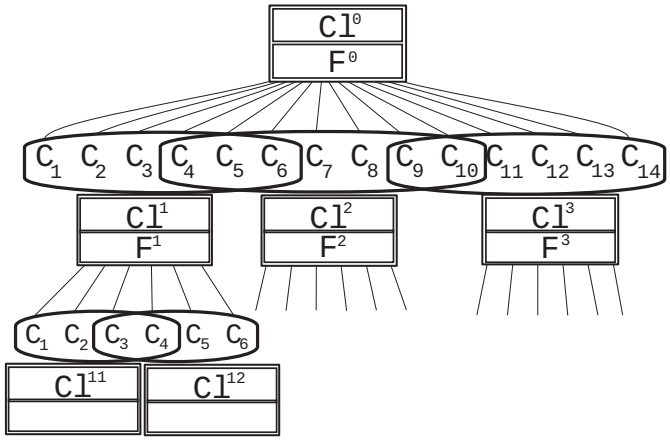
takimi, że

$$\begin{aligned}
sup(Cl^0(X)) &= \mathcal{C}, \\
\forall i \bigcup_{l:V^l \in child(V^i)} sup(Cl^l(X)) &= sup(Cl^i(X)),
\end{aligned}$$

gdzie $sup(Cl^i(X))$ określa suport Cl^i, czyli zbiór tych wszystkich klas, które mogą być rozpoznane przez Cl^i.

HCOC jest drzewem złożonym z węzłów V^i, z których każdy składa się z klasyfikatora Cl^i uczonego w procesie nauczania nadzorowanego oraz macierzy klastrowania F znajdowanej w procesie nienadzorowanego nauczania (klastrowania). Proces nauczania HCOC jest więc *fuzją* dwóch typów nauczania, w którym jeden stara się znaleźć najlepsze rozwiązanie problemu klasyfikacji dla pewnego podzbioru klas, a drugi wyszukuje pewne połączenie wyników nauczań osobnych podproblemów. Jak to zostanie pokazane, odpowiedni wybór heurystycznej funkcji kosztu pozwala na znalezienie optymalnej macierzy F, która minimalizuje całkowity błąd HCOC. Nauczanie HCOC jest związane z podejściem Expectation–Maximization EM, gdzie krok nauczania klasyfikatorów minimalizuje błędy w klastrach (Expectation w EM), klastrowanie natomiast przypisuje odpowiedzialności klastrom (Maximization) (Hastie *et al.*, 2001).

Funkcja *child* zwraca, dla węzła V^i, zbiór wszystkich jego potomków. Dla prostoty, wierzchołki będą numerowane w kolejności prefiksowej od lewej do prawej na każdym poziomie. Węzły V^i takie, że $|child(V^i)| > 0$, nazywamy *wewnętrznymi*, w przeciwnym wypadku *liśćmi*. Węzły w liściach składają się z samego klasyfikatora.



Rysunek 2.5. Przykładowa struktura HCOC. Węzły w liściach nie mają macierzy klastrowania F^i

Przykładowy HCOC jest pokazany na rysunku 2.5. Cały oryginalny problem składa się z przykładów etykietowanych przez 14 klas. Klasyfikator w korzeniu podejmuje próbę rozwiązania całego problemu.

Doświadczenie 2.8. *Dla wizualizacji podziału na klastry wykonane zostało doświadczenie dla problemu typu Mixture of Gaussians: na płaszczyźnie wylosowane zostały punkty z* 10 *klas o rozkładzie normalnym (po 500 przykładów uczących dla każdej klasy, ze stałą macierzą kowariancji). Rysunek 2.6 pokazuje zbiór uczący oraz klasyfikacje na kolejnych poziomach dla siatki* $1,000,000$ *przykładów testujących. HCOC jest ograniczony do ko-*

rzenia i dwóch poziomów poniżej. Klasyfikatory na kolejnych poziomach są słabe (użyte były sieci neuronowe z 2 neuronami ukrytymi uczone przez 20 *iteracji każda. Struktura utworzonego HCOC jest pokazana na rysunku 2.7.*

Na rysunku 2.8 zobrazowane jest działanie HCOC na pierwszym poziomie poniżej korzenia wraz z tworzeniem się klastrów. Dostrzegamy nakładanie się klastrów, tworzenie przez klastry podproblemów odpowiadających obszarom odpowiedzialności. Tu wszystkie ewaluacje wykonano metodą Restricted *(patrz rozdział 2.3.3). Każdy klaster ma obszar, dla którego związany z nim klasyfikator nie jest wywoływany, ponieważ w klasyfikatorze na poziomie wyżej aktywacje dla wszystkich klas z tego klastra mają wartości poniżej* $1/K$*. Mapy pokazują klasyfikacje według największej wartości aktywacji. Na ostatnim rysunku widać sumę ważoną tych klastrów i widać, że aktywacje niebędące zwycięzcami w klastrach mogą być końcowymi zwycięzcami. Na przykład klasa* C_0 *(kolor ceglasto-pomarańczowy) jest końcowym zwycięzcą na większym obszarze niż łącznie dla klastrów z osobna. Wynika to z faktu, że należy ona także do jednego klastra w którym nigdy nie jest zwycięzcą, gdyż jest mylona z innymi klasami, które przeważają.*

Na kolejnym rysunku 2.9 pokazany jest ten sam klasyfikator, ale z ewaluacją metodą α*-*Restricted*. W efekcie obszary, dla których klasyfikatory nie są wykorzystywane, są większe, ponieważ aktywacje muszą przekroczyć próg* $\alpha(K) > 1/K$ *(patrz rozdział 2.3.3). W obszarach, gdzie wskazanie prawidłowego klastra jest prostsze, tam odpowiedzialna jest mniejsza liczba klastrów. Na przykład, mimo że klasa* C_0 *należy do trzech klastrów, do jej wyboru wystarcza klaster* $Q_2 = \{C_0, C_3, C_4, C_6, C_8, C_9\}$*. Przy ewaluacji* Restricted *dla tej klasy aktywowany jest także klaster* $Q_3 = \{C_0, C_4, C_5, C_6, C_8\}$*. Wynika z tego, że dla* C_0 *wartości aktywacji aktywujących klasyfikator związany z* Q_3 *są mniejsze od* $\alpha(K)$ *(tutaj, dla* $K = 10$*,* $\alpha(10) = 0.180$*).*

Odpowiada to wyszukiwaniu przez model obszarów specjalizacji (ang. region of interest*, ROI Wołoszynski, Kurzyński, 2011; Itti, Baldi, 2005).*

2.2.1. Macierz klastrowania F

Binarna macierz klastrowania F opisuje klastry dla danego węzła V^i:

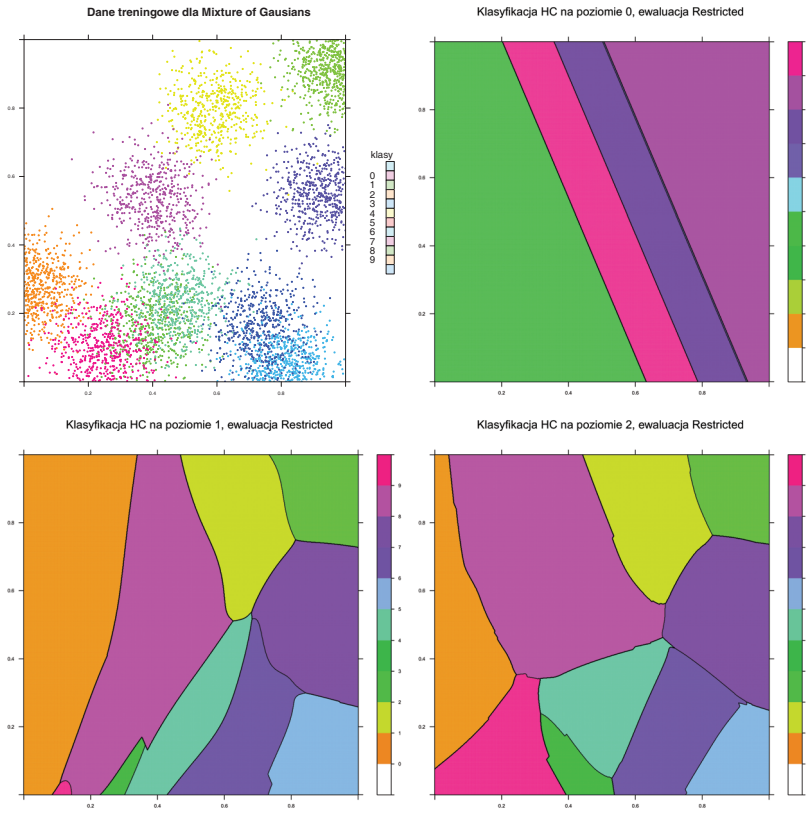
Definicja 2.9. Macierz klastrowania F^i węzła $V^i = (Cl^i, V^i)$ jest binarną macierzą

$$F^i = \{f^i_{kl}\}_{k=1,\ldots,|Q^i|;\ l=1,\ldots,|child(V^i)|} \tag{2.3}$$

taką, że $f^i_{kl} = 1 \ \ wtw \ \ C_k \in Q^l$, gdzie Q^l jest zbiorem klas rozpoznawanym przez potomka węzła V^i. W przeciwnym wypadku $f^i_{kl} = 0$.

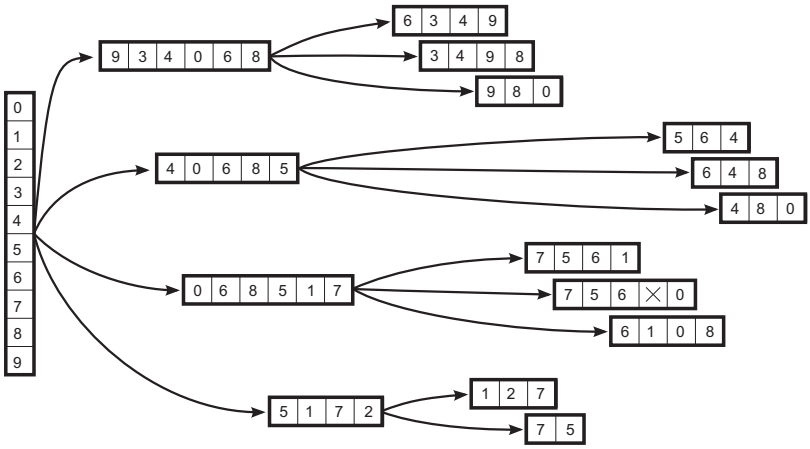
Postać macierzy F odzwierciedla charakterystykę jakości nauczania słabego klasyfikatora Cl^i w węźle V^i. Każda kolumna F odpowiada jednemu utworzonemu klastrowi, składającemu się z klas często przez Cl^i mylonych.

Hipoteza 2.10. *Słaby klasyfikator ma skłonność podawania jako odpowiedzi tych klas, które są silniejsze w zbiorze uczącym. Może to oznaczać, że są albo bardziej liczne, albo też typowe dla nich wektory atrybutów są podobne do typowych wektorów atrybutów innych klas.*



Rysunek 2.6. Działanie HCOC na syntetycznym problemie złożonym z 10 gausowskich chmur. Kolejne rysunki pokazują: wylosowane punkty, działanie klasyfikatora w korzeniu, w korzeniu i jednym poziomie w głąb, wszystkich 3 poziomach

Ponieważ klasyfikator jest słaby, ma on niewielką pojemność i jest w stanie rozpoznawać tylko niewielką część klas pojawiających się w zbiorze uczącym. Takie zachowanie sieci będzie odpowiadać skutecznemu, tzn. z niewielkim błędem, rozpoznawaniu *nie* przykładów z konkretnych pojedynczych klas, ale rozpoznawaniu *grup* klas. Będzie to realizowane przez konstruowany z nauczonego klasyfikator *uogólniony* (patrz rozdział 2.4.1). Można skorzystać z takiego zachowania i tanim kosztem zbudować klasyfikator (w rzeczywistości będzie on kombinacją już utworzonego klasyfikatora do klas), który z dużą skutecznością będzie rozpoznawać grupy klas. Rozpoznawaniem wewnątrz tej grupy zajmie się osobny klasyfikator. Będzie on miał prostszy problem do rozwiązania (Frank *et al.*, 2003; Hastie *et al.*, 2001).



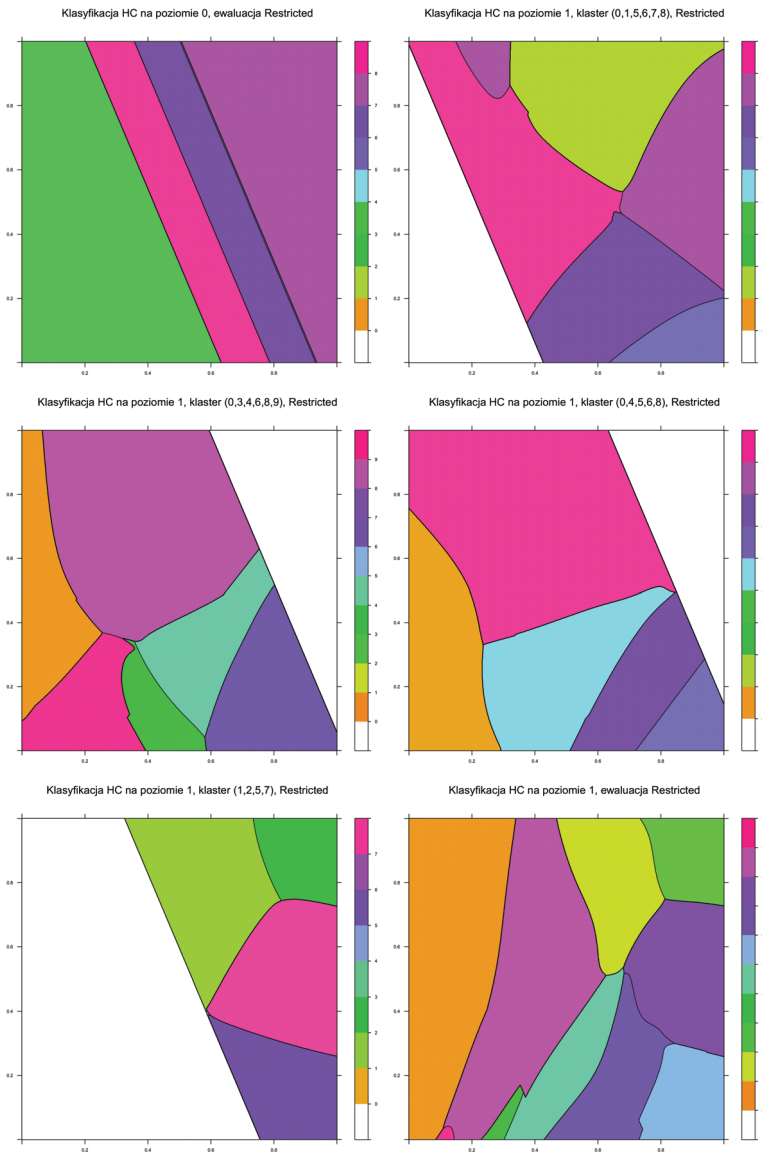
Rysunek 2.7. Struktura HCOC dla zadania Mixture of Gaussians pokazanego na rysunku 2.6. Klasy w klastrach zorganizowane tak, aby było widać nakładanie się klastrów w warstwach. Klasa o indeksie 1 nie należy do klastra w liściu z klasami $7, 5, 6, 0$, stąd znak $\times$

Poprawność macierzy F

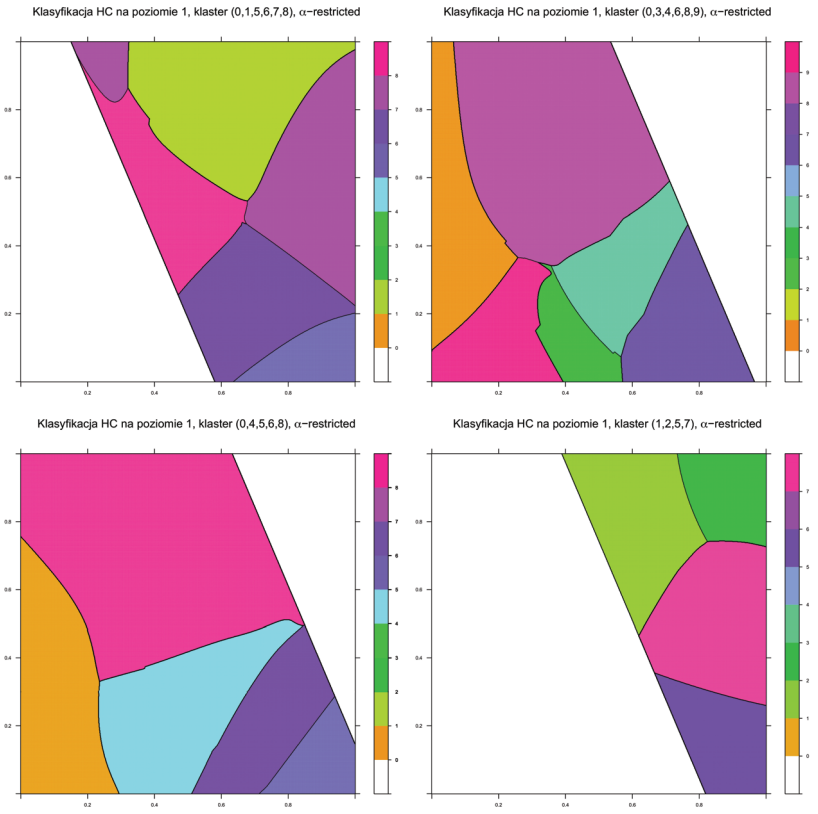
Macierz F będzie opisywać zachowanie się klasyfikatora w korzeniu poddrzewa HCOC:

- F ma tyle wierszy, ile klasyfikator uogólniony w korzeniu poddrzewa rozpoznaje klas,
- F ma tyle kolumn, na ile podproblemów został podzielony problem,
- w ramach każdego klastra są klasy, które często są klasyfikowane podobnie, czyli są z sobą mylone.

Liczba klastrów zależy w naturalny sposób od działania klasyfikatora w korzeniu. Celem HCOC jest taki podział zadania, aby po dodaniu każdego poziomu uzyskiwać zadania mniejsze i prostsze do rozwiązania. Związane jest z tym pojęcie *poprawności* macierzy klastrowania.



Rysunek 2.8. Problem Mixture of Gaussians i podział klastrów w korzeniu. Kolejno: aktywacja w korzeniu, aktywacje dla klastrów na poziomie poniżej korzenia. Ewaluacja typu Restricted. Obszary są niepokolorowane w miejscach, gdzie aktywacja w korzeniu poniżej *prior* $P(C_i)$ powoduje nieewaluowanie klastra. Na końcu mapa połączonych wyników pierwszego poziomu HCOC



Rysunek 2.9. Problem Mixture of Gaussians i podział klas w korzeniu. Wszystkie ewaluacje metodą α-RESTRICTED

Definicja 2.11. Macierz klastrowania F jest poprawna, jeśli spełnia następujące warunki

$$\forall k \in sup(Cl^i)\ \exists l: \quad f_{kl} = 1, \tag{2.4}$$

$$\forall k \notin sup(Cl^i): \quad f_{kl} = 0\ l = 1, \ldots, L, \tag{2.5}$$

$$\forall l: \quad 2 \leq \sum_{k \in sup(Cl^i)} f^i_{kl} < |sup(Cl^i)|, \tag{2.6}$$

$$\forall l_1, l_2: \quad \left(\forall k \in sup(Cl^i)\ f^i_{kl_1} \geq f^i_{kl_2}\right) \Longrightarrow l_1 = l_2, \tag{2.7}$$

$$\exists l_1, l_2,\ l_1 \neq l_2: \quad \sum_{k \in sup(Cl^i)} f^i_{kl_1} f^i_{kl_2} > 0\ . \tag{2.8}$$

W szczególnych wypadkach warunki te mogą być osłabione (pominięte), co wymaga jednak zmiany definicji drzewa HCOC. W szczególności, ważne są następujące komentarze:

- warunek (2.4) opisuje żądanie, aby każda klasa była rozpoznawana przez co najmniej jeden klasyfikator pochodny (a klasa należała do jego suportu); można sobie jednak wyobrazić sytuację, że dana klasa C_k jest rozpoznawana przez aktualny *Cl* bezbłędnie i nie potrzebuje być łączona z innymi; wymaga to drobnej modyfikacji algorytmu łączenia wyników klasyfikatorów w węzłach;

- jeśli przykłady z danej klasy są rozpoznawane bezbłędnie, to nie ma konieczności łączenia jej z inną klasą; taka klasa może być rozpoznawana bez przechodzenia w dół drzewa HCOC, co wymaga także osłabienia warunku (2.6) opisującego żądanie tak, żeby każdy klaster składał się z co najmniej dwóch klas;

- warunek (2.7) wymaga, aby żaden klaster nie był podzbiorem innego, oraz aby klastry były różne;

- warunek (2.8) wymaga, aby co najmniej dwa klastry nakładały się przynajmniej jedną klasą; warunek ten może być rozszerzony do żądania, żeby *każdy* klaster potomny nakładał się z każdym innym (gdy oba są pochodnymi innego klastra nadrzędnego) lub też żeby każda klasa należała do co najmniej dwóch klastrów; może poprawić dokładność, jednak zwiększy złożoność obliczeniową;

- dana klasa C_k *może* należeć do wszystkich klastrów, co jest jednak sytuacją rzadką i nie sprzyja skutecznemu podziałowi, jeśli dzieje się tak dla więcej niż jednej z klas; taka sytuacja może mieć miejsce, gdy liczba przykładów uczących dla C_k jest znacznie większa niż dla innych klas.

Macierz F może przypominać w pewnym stopniu podejście ECOC – *Error-Correcting Output Codes* (Dietterich, Bakiri, 1995; Tapia *et al.*, 2010). W modelu ECOC dla klasyfikacji K klas budowany jest ciąg L klasyfikatorów binarnych w następujący sposób: w każdym etapie zbiór wszystkich klas jest dzielony losowo na części A_j i B_j i etykietowany binarnie 0 i 1; dla takiego podziału tworzony jest klasyfikator typu *grupa-klas-przeciwko-reszcie*. Przy klasyfikacji przykład jest rozpoznawany przez wszystkie klasyfikatory binarne. Jeśli przykład jest klasyfikowany jako należący do B_j, to *wszystkie* klasy należące do B_j dostają pojedynczy głos. Na końcu wybierana jest klasa z najwyższą liczbą głosów.

Odpowiednikiem klastra jest w ECOC zbiór wszystkich klas rozpoznawanych przez j-ty klasyfikator jako należące do B_j. Różnica polega na tym, że klastry w ECOC są generowane losowo, a w HCOC polegają na analizie poprawności i pomyłek klasyfikatora. HCOC wydobywa z klasyfikatora więcej wiedzy. Te podejścia są wyraźnie różne, mimo pozornego podobieństwa macierzy ECOC i F.

Liczba poprawnych klastrowań

Określenie liczby różnych macierzy klastrowania F spełniających powyższe warunki jest problemem złożonym dla dowolnie ustalonych liczby klas K i liczby klastrów L. Jest to rodzaj problemu Dedekinda określenia liczby wszystkich monotonicznych funkcji Boolowskich n zmiennych, które mogą być zrealizowane wyłącznie przy wykorzystaniu operacji AND i OR (Podolak *et al.*, 2012b).

Lemat 2.12. *Dla K klas i $L = 3$ klastrów liczba poprawnych klastrowań $\vartheta(K,L)$ dla $K \geq 3$ wynosi*

$$\vartheta(3,3) = 1,$$
$$\vartheta(K+1,3) = 7\vartheta(K,3) + 2 \cdot 7^K - 3 \cdot 4^K + \frac{4K-11}{6} \cdot 3^K - \frac{15K-16}{4} \cdot 2^K + 9K - \frac{1}{2}.$$

Dla kilku początkowych K wartości $\vartheta(K,3)$ przedstawione są w tabeli 2.1. Jak widać, są one zbyt duże, aby móc je przeglądnąć w poszukiwaniu najlepszego podziału.

Tabela 2.1. Liczba poprawnych klastrowań dla 3 klastrów

K	3	4	5	6	7	8	9	10
$\vartheta(K,3)$	1	38	675	7 840	74 291	630 546	5 014 843	38 290 580

2.3. Ewaluacja klasyfikatora HCOC

W dodatku A.6 przedstawione zostały różne modele składania wyników w komitetach maszyn. Model HCOC jest modelem *pośrednio zależnym* od danych, w których wagi składowych klasyfikatorów są *zależne* od przykładu aktualnie klasyfikowanego i końcowa funkcja klasyfikacji ma postać

$$f(w_i(Cl^j(x)), Cl^i(x)).$$

Dzięki temu dla każdego przykładu z osobna wyliczany jest, pod postacią wagi, wpływ każdego klasyfikatora bazowego. Odpowiada to mierze *kompetencji* tego klasyfikatora i rozpoczynającego się od niego poddrzewa (Wołoszynski, Kurzyński, 2011; Brodowski, Podolak, 2011; Dara *et al.*, 2009).

2.3.1. Agregacja wyników klasyfikacji w modelach złożonych

Istnieje wiele wykorzystywanych metod selekcji końcowej klasy (kilka podstawowych jest wymienione w dodatku A.7). HCOC wykorzystuje ważony model me-

diany, w którym klasyfikacje każdego Cl^l są dodatkowo ważone przez odpowiednią wagę w_l

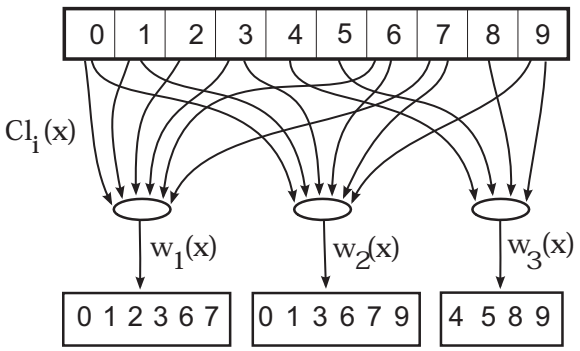
$$c = \arg\max_{C_k} \sum_{l=1}^{L} w_l(x) P(C_k|Cl^l, x). \tag{2.9}$$

Jednocześnie, ponieważ HCOC jest modelem pośrednio zależnym od danych, waga jest zależna od klasyfikowanego przykładu $w_l = w_l(x)$. Konieczne jest takie zdefiniowanie wag, żeby zmaksymalizować prawdopodobieństwo prawidłowego wyboru.

Ważnym elementem budowy systemu jest takie zdefiniowanie wag, aby dla danego klasyfikowanego przykładu x klastry zawierające prawdziwą klasę miały wagę relatywnie *wyższą* niż wagi klastrów nie zawierających tej klasy. Jeśli to zostanie zapewnione, to można oczekiwać wysokiej skuteczności działania modelu.

2.3.2. Wagi klastrów HCOC

W modelu HCOC podstawowym postulatem jest podział w węźle $V^i = (Cl^i, F^i)$ na podproblemy złożone z L^i klastrów $Q^{i1}, \ldots, Q^{il}, \ldots, Q^{iL^i}$, które nakładają się. Podział następuje na podstawie wyników nauczania klasyfikatora Cl^i poprzez znalezienie, które klasy $C_j, C_k \in Q^i$ są często podobnie klasyfikowane. Tak więc wagi każdego z węzłów pochodnych są zależne od wektora aktywacji dla aktualnie klasyfikowanego przykładu x. Jest to pokazane na rysunku 2.10.



Rysunek 2.10. Przypisanie przykładów do klas. Z tych aktywacji będą wynikać wagi dla aktualnego przykładu

Podejście standardowe obliczania wag klastrów

Definicja 2.13. Niech $V^i = (Cl^i, F^i)$. Waga klastrów klastrów jest funkcją $w^i : \mathcal{X} \longrightarrow [0,1]^{L^i}$, gdzie $w^i(x) = [w^i_1(x), \ldots, w^i_{L^i}(x)]$

$$
\begin{aligned}
w^i_l(x) &= \frac{\sum_{k \in Q^i_l} Cl^i_k(x)}{\sum_{m:V^m \in child(V^i)} \sum_{k \in Q^i_m} Cl^i_k(x)} \\
&= \frac{\sum_{k=1}^{K} f^i_{kl} Cl^i_k(x)}{\sum_{m:V^m \in child(V^i)} \sum_{k=1}^{K} f^i_{kl} Cl^i_k(x)}, \qquad l = 1, \ldots, L^i \\
&= \frac{\sum_{k=1}^{K} f^i_{kl} Cl^i_k(x)}{\sum_{l'=1}^{L^i} \sum_{k'=1}^{K} f^i_{k'l'} Cl^i_{k'}(x)}, \qquad l = 1, \ldots, L^i. \qquad (2.10)
\end{aligned}
$$

Mianownik (2.10) normalizuje wyrażenie, ponieważ ze względu na nakładanie się klas suma wag $\sum_{l=1}^{L} w^i_l(x) \geq 1.0$.

Taka definicja uwzględnia założenie, że Cl^i jest słaby: klasyfikuje mało dokładnie, jednak robi to lepiej niż losowo. W związku z tym aktywacje $Cl^i_j(x)$ niosą z sobą pewną, choć niedokładną, informację o prawdopodobieństwie, że przykład x należy do C_j. Jeśli więc $C_j \in Q^l$, to ta informacja składa się na wagę w_l. Im wyższa jest wartość $Cl_j(x)$, tym bardziej Cl wskazuje na klaster Q^l (i wszystkie inne, do których C_j należy) i tym wyższa jest waga w_l. W rozdziale 2.3.4 pokazane będzie, że dla danego przykładu klastry rozpoznające jego prawdziwą klasę mają zdecydowanie *wyższe* wagi niż klastry nie zawierające tej klasy.

Uwaga 2.14. *Waga klastrów węzła V^i jest zdefiniowana jako funkcja $w^i : \mathcal{X} \longrightarrow [0,1]^{I^i}$, a więc pochodząca z całej przestrzeni wejściowej $\mathcal{X}$, nie przestrzeni $\mathcal{X}_i$ ograniczonej do klas rozpoznawanych przez V^i. Odpowiada to warunkowi, że wagi są obliczane w trakcie ewaluacji dla przykładu x, którego klasa jest nieznana, a nie są ustalane na stałe na etapie nauczania.*

Podejście stochastyczne Γ

Standardowy algorytm obliczania wag $w_l(x)$ zakłada, że aktywacja $Cl^i_j(x)$ klasyfikatora nadrzędnego przybliża prawdopodobieństwo, iż prawdziwą jest klasa C_j. Trzeba jednak uwzględnić, że Cl^i popełnia szereg błędów. Te błędy mogą być dwóch typów

- *przypadkowych*, polegających na niepoprawnej klasyfikacji wektora atrybutów x, przy czym błąd ten jest zależny tylko od x i słabości klasyfikatora Cl: na ten rodzaj błędów niewiele da się poradzić (poza, oczywiście, ulepszeniem klasyfikatora Cl),
- *systematycznych*, polegających na wielokrotnym myleniu przez Cl odpowiedzi dla par klas C_j i C_k: informacje o tym typie błędów są dostępne dzięki analizie macierzy błędnej klasyfikacji M.

Niech klasyfikator Cl^i klasyfikuje wektor atrybutów x, przypisując go do klasy C_j (oznaczone tutaj, dla prostoty, jako $pr(x) = j$). Prawdopodobieństwo, że prawdziwą klasą x jest C_t (oznaczone jako $tr(x) = t$), można, z prawa Bayesa, ewaluować jako

$$\begin{aligned} P(tr(x) = t|pr(x) = j) &= \frac{P(pr(x) = j|tr(x) = t)P(tr(x) = t)}{\sum_{k=1}^{K} P(pr(x) = j|tr(x) = k)P(tr(x) = k)} \\ &= \frac{m_{tj}p_t}{\sum_{k=1}^{K} m_{kj}p_k}, \end{aligned} \tag{2.11}$$

gdzie $P = [p_1, \ldots, p_K]$ jest wektorem prawdopodobieństw *a priori* dla wszystkich klas, które są obliczone ze zbioru uczącego $p_t = |\{(x, C_t) \in \mathcal{D}\}|/|\mathcal{D}|$.

Niech $\Gamma^i = \Gamma^i(M^i, P) = (\gamma^i_{jk}),\ j,k = 1, \ldots, K$ oraz

$$\gamma^i_{jk} = \frac{m^i_{jk}p_j}{\sum_{s=1}^{K} m^i_{sj}p_s}. \tag{2.12}$$

To prowadzi do nowej, stochastycznej, definicji wag:

Definicja 2.15. Niech $V^i = (Cl^i, F^i)$. Stochastyczna Γ waga klastrów, uwzględniająca błędy systematyczne, jest funkcją $w^i : \mathcal{X} \longrightarrow [0,1]^{L^i}$

$$w^i_l(x) = \frac{\sum_{k=1}^{K} f^i_{kl}\gamma^i_{kj}}{\sum_{l'}^{L^i}\sum_{k'}^{K} f^i_{k'l'}\gamma^i_{k'j}}, \tag{2.13}$$

gdzie $j = \arg\max_k Cl^i_k(x)$ jest indeksem klasy, którą klasyfikator Cl^i by wybrał, gdyby był końcowym klasyfikatorem.

Niech $pr(x)$ będzie klasą przewidywaną przez Cl^i jako najbardziej prawdopodobna dla x. W podstawowym podejściu, we wzorze (2.10), wszystkie wartości wektora aktywacji $Cl^i(x)$ są wykorzystywane bezpośrednio dla obliczenia wagi klastrów.

Podejście stochastyczne jest oparte na wzorze Bayesa i bierze pod uwagę tylko indeks największej wartości wektora aktywacji Cl^i (czyli $pr(x)$). Można interpretować $pr(x) = C_j$ jako zdarzenie losowe i z twierdzenia Bayesa wyliczyć $P(tr(x) = C_t|pr(x) = C_j)$ (równanie (2.11)), a następnie wartości aktywacji Cl^i_k zamienić, wykorzystując znormalizowaną postać γ_{kj} dla $j = \arg\max_k Cl^i_k(x)$.

Porównania wyników klasyfikacji przy użyciu standardowej i stochastycznej Γ ewaluacji wag podane są w tabeli 2.2. Wyniki obu podejść są zwykle podobne.

2.3.3. Ewaluacja poddrzew

Ewaluacja przykładów dla nauczonego HCOC następuje metodą z góry do dołu: najpierw obliczana jest aktywacja klasyfikatora Cl^0 w korzeniu, z kolei wyliczany

jest wektor wag wszystkich węzłów (i związanych z nimi klastrów) pochodnych od korzenia $w^0 = [w_1^0, \ldots, w_{L^0}^0]$. Następnie wszystkie poddrzewa traktowane są jak osobne klasyfikatory HCOC. Po dojściu do liści następuje obliczenie końcowej aktywacji $y^{HCOC}(x)$ jako kolejnych kombinacji liniowych

$$y^{HCOC}(x) = \sum_{l \in child(V^0)} w_l(x)^0(x) y^l(x), \tag{2.14}$$

gdzie $y^l(x)$ jest aktywacją dla przykładu x poddrzewa HCOC, które ma swój korzeń w węźle V^l. Istnieje kilka możliwych podejść do tej ewaluacji.

Tabela 2.2. Porównania wyników klasyfikacji przy użyciu standardowej (std) i stochastycznej (Γ) ewaluacji wag. Użyte metody klastrowania to, $clHC_G$ – klastrowanie Bayesowskie ze śladem macierzy generalizacji jako wartością dopasowania, $clHC_R$ – klastrowanie Bayesowskie z ewaluacją ryzyka HCOC z twierdzenia 2.31, GA_1 – klastrowanie z wykorzystaniem algorytmu genetycznego i ewaluacją ryzyka HCOC jako funkcją dopasowania, GA_2 – genetyczne z kombinacją śladu macierzy generalizacji i ewaluacją ryzyka HCOC jako funkcją dopasowania. Dla ewaluacji użyty błąd $Err^{0.632}$ (patrz A.13). Najlepsze wyniki wytłuszczonym drukiem. Wyniki dla 100 powtórzeń zbioru *bootstrap* dla dwupoziomowego HCOC

	Błąd $Err^{(.632)}$							
Funkcja kosztu →	$1 - Cl_{tr(x)}^{HCOC}$				$\ell_{0/1}$			
Problem	$clHC_G$	$clHC_R$	GA_1	GA_2	$clHC_G$	$clHC_R$	GA_1	GA_2
Vowel Γ	0.366	0.390	**0.345**	0.360	0.246	0.240	**0.226**	0.267
Vowel std	**0.331**	0.353	0.338	0.339	0.240	0.232	**0.222**	0.264
Primary-tumor Γ	0.610	**0.609**	0.618	0.617	0.471	**0.459**	0.473	0.524
Primary-tumor std	**0.599**	0.601	0.616	0.609	0.467	**0.455**	0.471	0.517
Arrhythmia Γ	0.619	0.614	**0.547**	0.594	0.461	0.460	0.466	**0.459**
Arrhythmia std	0.619	0.613	**0.547**	0.596	0.460	0.459	0.465	**0.458**
Audiology Γ	0.439	**0.434**	0.480	0.496	0.277	**0.253**	0.294	0.362
Audiology std	0.431	**0.425**	0.477	0.493	0.274	**0.246**	0.293	0.363
Zoo Γ	0.103	0.114	**0.096**	0.111	0.068	0.072	**0.067**	0.085
Zoo std	0.102	0.113	**0.099**	0.110	0.068	0.070	**0.066**	0.084

Pełne obliczenie wszystkich poddrzew All-subtrees

Definicja 2.16. W podejściu All-subtrees węzeł V^i zwraca wektor prawdopodobieństw klas $y^i(x) = [y^i_1(x), \ldots, y^i_K]$ zdefiniowanym jako
(2.15)

$$y^i_j(x) = \begin{cases} Cl^i_j(x) & \text{jeśli } V^i \text{ jest liściem,} \\ \sum_{\substack{l: V^l \in child(V^i) \\ C_j \in Q^i_l}} w^i_l(x) y^l_j(x) & \text{jeśli } V^i \text{ jest węzłem wewnętrznym,} \\ 0 & \text{jeśli } C_j \notin Q^i, \end{cases}$$

gdzie $j = 1, \ldots, K$.

Takie standardowe podejście bierze pod uwagę wszystkie możliwe poddrzewa. Przy założeniu, że aktywacja korzenia niesie informacje, i przy zdefiniowanym sposobie obliczania wag w_l często będzie występować sytuacja, że jeden (lub więcej) klastrów *nie* zawierających prawdziwej klasy rozpoznawanego przykładu będzie mieć bardzo niską wagę, czyli prawdopodobieństwo $P(tr(x) \in Q^j|x)$ będzie niskie. Mimo to uwzględnienie go spowoduje zaszumienie informacji poprzez użycie tej, która jest z wysokim prawdopodobieństwem nieprawdziwa. Stąd metodę All-subtrees należy stosować z ostrożnością.

Ewaluacja wzdłuż jednej ścieżki Single-path

Definicja 2.17. W podejściu Single-path węzeł V^i zwraca wektor prawdopodobieństw klas $y^i(x) = norm([y^i_1(x), \ldots, y^i_K])$ zdefiniowanym jako

(2.16)

$$y^i_j(x) = \begin{cases} y^l_j(x), \text{ gdzie } l = \arg\max_{l'} w^i_{l'}(x) & \text{jeśli } V^i \text{ jest wewnętrznym węzłem, oraz } C_j \in Q^i_l, \\ Cl^i_j(x) & \text{jeśli } V^i \text{ jest liściem, albo } C_j \notin Q^i_l, \\ 0 & \text{jeśli } C_j \notin Q^i, \end{cases}$$

$j = 1, \ldots, K$, gdzie $norm(v)$ jest funkcją normalizującą.

To podejście jest przeciwieństwem All-subtrees, gdyż ewaluowana jest jedynie jedna ścieżka z węzła do jednego liścia w drzewie. Odpowiada to, na każdym poziomie drzewa HCOC, wyborowi $\arg\max_l P(x \in Q^l|x)$, czyli najbardziej prawdopodobnemu klastrowi. Dla klas, dla których Cl^i jest stosunkowo dobrze nauczony, to rozwiązanie daje poprawne wyniki, jednak najczęściej są to wyniki najsłabsze. Dzieje się tak, ponieważ Single-path nie wykorzystuje w pełni zalety HCOC, którą jest nakładanie się klastrów.

Ewaluacja ograniczona Restricted

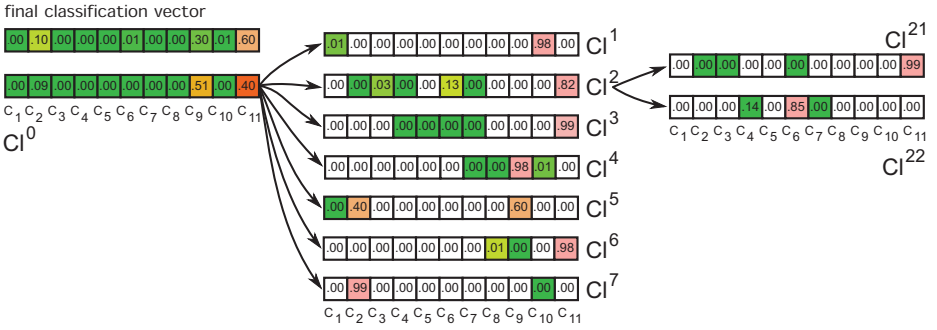
All-subtrees i Single-path są podejściami ekstremalnymi. Należy zauważyć, że jeśli klasyfikator Cl^i byłby losowy, to wartość oczekiwana aktywacji (sumy akty-

wacji są znormalizowane) $E[Cl_j(x)|x]$ będzie równa prawdopodobieństwu *a priori* klasy C_j. Można więc założyć, że jeśli $Cl_j(x) < P(C_j)$, to zachodzi małe prawdopodobieństwo, aby to była właściwa klasa, i, wobec tego, ominąć tę aktywację przy obliczaniu wag. Pozwoli to wykluczyć klastry, dla których aktywacja żadnej z tworzących je klas nie jest wystarczająco prawdopodobna. To rozumowanie daje asumpt do zdefiniowania metody ewaluacji RESTRICTED.

Definicja 2.18. W podejściu RESTRICTED węzeł V^i zwraca wektor prawdopodobieństw klas $y^i = norm([y^i_1, \ldots, y^i_K])$ zdefiniowanym jako

$$y^i_j(x) = \begin{cases} \sum_{\substack{l:V^l \in child(V^i) \\ C_j \in Q^i_l \\ \exists k: C_k \in Q^i_l \wedge Cl^i_k(x) \geq mb(C_k)}} w^i_l(x) y^l_j(x) & \text{jeśli } V^i \text{ jest węzłem wewnętrznym,} \\ Cl^i_j(x) & \text{jeśli } V^i \text{ jest liściem,} \\ 0 & \text{jeśli } C_j \notin Q^i, \end{cases} \tag{2.17}$$

gdzie $j = 1, \ldots K$, a $mb(C_k)$ jest *miarą ufności* (ang. *measure of belief*) (Giarratano, Riley, 1994) klasy C_k jako prawdopodobieństwa *a priori* klasy C_k. Wartości $mb(C_k)$ są aproksymowane ze zbioru uczącego (Podolak, 2008) jako $mb(C_k) = p_k = |\{(x_i, C_k) \in D\}|/|D|$.



Rysunek 2.11. Ewaluacja metodą ALL-SUBTREES. Kwadraty zaznaczone kolorem oznaczają klasy rozpoznawane w danym klastrze. Po lewej stronie u dołu aktywacja klasyfikatora w korzeniu, u góry końcowa aktywacja całego modelu HCOC

Na rysunku 2.11 pokazana jest przykładowa ewaluacja przykładu x metodą ALL-SUBTREES. Korzeń Cl^0 ma niezerowe aktywacje jedynie dla dwóch klas C_9 (0.51) i C_{11} (0.40). Klasyfikatory Cl^4 i Cl^5 na następnym poziomie rozpoznają klasę C_9, ale nie C_{11}. Cl^2 i Cl^3 rozpoznają C_{11}, ale nie C_9. Dzięki temu i korelacjom z innymi klasami końcowa klasyfikacja jest (prawidłowo) zmieniona na C_{11}. Wiele z klasyfikatorów, np. Cl^6 dla klasy C_8, ma minimalne wartości aktywacji, które z bardzo wysokim prawdopodobieństwem *nie wskazują* na poprawną klasę, wprowadzając jedynie szum. Podejście RESTRICTED nie uwzględnia tych aktywacji.

Uwaga 2.19. *Niech aktywacja dla danej klasy C_i należącej do klastra Q będzie niższa od prior $P(C_i)$. Jeśli jednak aktywacja co najmniej jednej innej klasy $C_k \in Q$ będzie większa od $P(C_k)$, to klaster Q i tak będzie ewaluowany w podejściu* Restricted*, a klasa C_i zwrócona jako odpowiedź, jeśli to ona jest prawdziwa. Ma to znaczenie, gdy różnice wartości aktywacji różnią się niewiele od prior $P(C)$.*

Ewaluacja ograniczona uwzględniająca słabość klasyfikatorów α-Restricted

Metoda Restricted w dalszym ciągu nie wykorzystuje jednak założenia przyjętego przy budowie HCOC, że klasyfikatory bazowe w węzłach są *co najmniej* słabe. Według definicji 1.17 oznacza to, że dla K-klasowego klasyfikatora wartość oczekiwana aktywacji dla prawdziwej klasy wynosi co najmniej $\alpha(K)$: $E[Cl_j(x)|tr(x) = C_j] \geq \alpha(K)$. Prowadzi to do modelu ewaluacji α-Restricted.

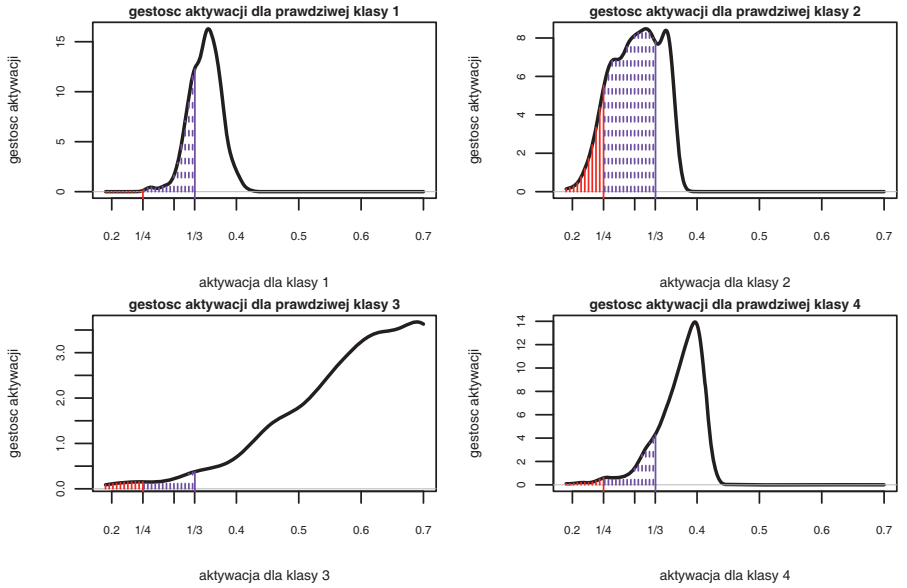
Definicja 2.20. Niech węzeł $V^i = (Cl^i, F^i)$ będzie węzłem z K-klasowym klasyfikatorem Cl^i. W podejściu α-Restricted węzeł V^i zwraca wektor prawdopodobieństw klas $y^i(x) = norm([y^i_1(x), \ldots, y^i_K(x)])$ zdefiniowanym jako
(2.18)

$$y^i_j(x) = \begin{cases} \sum\limits_{\substack{l:V^l \in child(V^i) \\ C_j \in Q^i_l \\ \exists k: C_k \in Q^i_l \wedge Cl^i_k(x) \geq \alpha(K))}} w^i_l(x) y^l_j(x) & \text{jeśli } V^i \text{ jest wewnętrzny} \\ & \text{i } \exists k \leq K\ Cl^i_k(x) > \alpha(K), \\ \sum\limits_{\substack{l:V^l \in child(V^i) \\ C_j \in Q^i_l \\ \exists k: C_k \in Q^i_l \wedge Cl^i_k(x) \geq mb(C_k))}} w^i_l(x) y^l_j(x) & \text{jeśli } V^i \text{ wewnętrzny} \\ & \text{i } \forall k \leq K\ Cl^i_k(x) < \alpha(K), \\ Cl^i_j(x) & \text{w przeciwnym wypadku,} \end{cases}$$

gdzie $j = 1, \ldots K$, gdzie $mb(C_k)$ jest miarą ufności C_k aproksymującej prawdopodobieństwo *a priori* $P(C_k)$.

Według metody α-Restricted wybierane są tylko te klastry potomne, w których jest co najmniej jedna klasa C_k o aktywacji wyższej od $\alpha(K)$. W Restricted taki warunek (względem $mb(\cdot)$) był wystarczający, ponieważ ze względu na normalizację wektora aktywacji co najmniej jedna klasa musiała mieć aktywację większą lub równą $P(C_k)$. W przypadku α-Restricted może zajść sytuacja, w której *wszystkie* klasy mają aktywację *niższą* od $\alpha(K)$ – wtedy używana jest metoda Restricted.

Na rysunku 2.12 pokazane są empiryczne gęstości aktywacji klas nauczonych w pewnym syntetycznym czteroklasowym problemie. Powierzchnia czerwonego obszaru poniżej wartości oczekiwanej $E[P(C_j)]$ odpowiada oczekiwanemu błędowi popełnianemu przy wyborze klastrów zawierających daną klasę. Obydwa obszary, czerwony i niebieski, odpowiadają błędowi popełnianemu przy wyborze klastrów zawierających klasy, których aktywacja jest poniżej $\alpha(K)$. Pokazuje to wyraźnie, jak wiele można zyskać w stosunku do metody All-subtrees przez wybór metody Restricted albo, jeszcze lepiej, α-Restricted. Tabela 2.3 podaje wartości



Rysunek 2.12. Wykresy empirycznych gęstości aktywacji $Cl_j(x)$ dla $x \in C_j$ słabego klasyfikatora. Na czerwono zaznaczony jest obszar poniżej $P(C_i)$ dla każdej z klas, na niebiesko obszar powyżej prawdopodobieństwa *a priori* i poniżej $\alpha(4) = 1/3$. Wartości powierzchni podane są w tabeli 2.3

Tabela 2.3. Frakcja powierzchni pod krzywą empirycznych gęstości rozkładów aktywacji 4 klas z rysunku 2.12

	Klasa 1	Klasa 2	Klasa 3	Klasa 4
$Cl(x) \leq P(C_j)$	0.00064	0.11543	0.00822	0.01492
$P(C_j) \leq Cl(x) \leq \alpha(K)$	0.25914	0.62606	0.01839	0.13728
$Cl(x) \leq \alpha(K)$	0.25978	0.74149	0.02662	0.15221

powierzchni pod krzywą dla obydwu obszarów. Widać, że jeśli w trakcie klasyfikacji opuści się rozpoznawanie przykładów nieosiągających aktywacji *prior* $P(C)$ bądź $\alpha(K)$, tak jak czynią, odpowiednio, RESTRICTED i α-RESTRICTED, możliwe jest zmniejszenie błędów. Potwierdzają to zresztą wszystkie wykonane doświadczenia. Różnice nie będą jednak tak duże, ponieważ część klas mimo małej aktywacji w dalszym ciągu jest brana pod uwagę jako elementy klastrów z klasą o wystarczająco wysokiej aktywacji. Zabezpiecza to przed błędnym działaniem, gdy żaden klaster nie zawierający poprawnej klasy nie będzie dalej ewaluowany.

2.3.4. Zależność wag klastrów od składowych klas

Zgodnie ze wzorem (2.14) końcowa aktywacja HCOC jest obliczana jako suma ważona aktywacji klasyfikatorów bazowych następnego poziomu. Wagi obliczane są zgodnie z wzorem (2.10) bądź (2.13). Przy takiej definicji HCOC zwraca poprawne wyniki, jeśli, obliczone na podstawie aktywacji klasyfikatora w korzeniu danego poddrzewa, wagi są wysokie dla klastrów, które *zawierają* prawdziwą klasę danego przykładu, oraz niskie dla klas *nie* zawierających tej klasy.

Podproblemy są tworzone przez analizę ujawniającą, które przykłady są często z sobą mylone, czyli które wartości m_{ij} są wysokie dla prawdziwej klasy C_i mylonej z C_j. Odpowiada to następującemu procesowi:

1. niech Cl będzie słabym klasyfikatorem, w którym wektory aktywacji X losowane są z K-wymiarowego rozkładu Dirichleta o parametrach $(1, \ldots, 1)$ (patrz A.12),

2. każdy wektor aktywacji jest sortowany od wartości największej do najmniejszej (oznaczymy ten wektor jako $[X_{(1)}, \ldots, X_{(K)}]$),

3. ułamek $\sum_{i=1}^{p} X_{(i)} / \sum_{j=1}^{p} X_j$ dla $p = 1, \ldots, K$ pokazuje stosunek wag klastrów zawierających najwyższe aktywacje do równolicznych klastrów wybranych losowo.

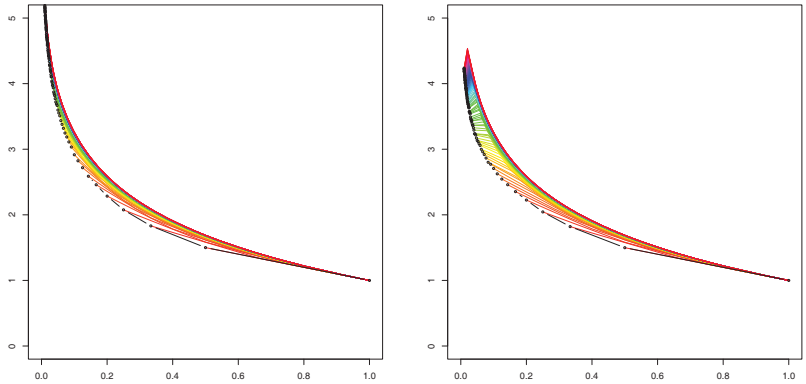
Stosunek ten jest pokazany na rysunku 2.13. Widać, że dla takiego teoretycznego procesu i liczby klas nie większej niż $K/2$ ten stosunek jest większy od 2.

Jeśli założymy, że Cl jest słaby, a aktywacje dla pozostałych klas są generowane z $(K-1)$-wymiarowego rozkładu Dirichleta (założenie analogiczne do tego w dowodzie twierdzenia o słabym klasyfikatorze 1.18), interesować nas będzie stosunek

$$(X_{tr} + \sum_{\substack{i=1 \\ (i)\neq tr}}^{p} X_{(i)}) / (\sum_{j=1}^{p} X_j), \tag{2.19}$$

dla $p < K$. Symulacja tego procesu jest także pokazana na rysunku 2.13. Potwierdza to sposób dzielenia problemu na mniejsze podproblemy składające się z często mylonych klas: są one łączone w jednym klastrze, gdy aktywacje dla nich często są wysokie dla tych samych przykładów. Jednocześnie powoduje to, że klastry zawierające klasę prawidłową mają wysoką aktywację.

Taki proces zakłada jednak pesymistycznie, że aktywacje dla wszystkich klas poza prawdziwą są rozłożone równomiernie, co nie jest zwykle prawdą dla rzeczywistych problemów. Jak pokazano w rozdziale 2.1, aktywacje słabo nauczonego klasyfikatora układają się według schematu: aktywacja dla klasy prawdziwej, następnie kilka wyższych aktywacji dla innych klas, a większość pozostałych ma



Rysunek 2.13. Po lewej stosunek sumy p najwyższych aktywacji do p aktywacji losowych dla K-elementowego procesu (dla symulacji wykorzystano dane z problemu vowel). Po prawej stosunek sumy aktywacji dla prawdziwej klasy (przy założeniu, że klasyfikator jest słaby) oraz $p-1$ innych największych aktywacji do sumy p aktywacji wybranych losowo. Oś pozioma pokazuje stosunek liczby klas w klastrze p do liczby wszystkich klas K

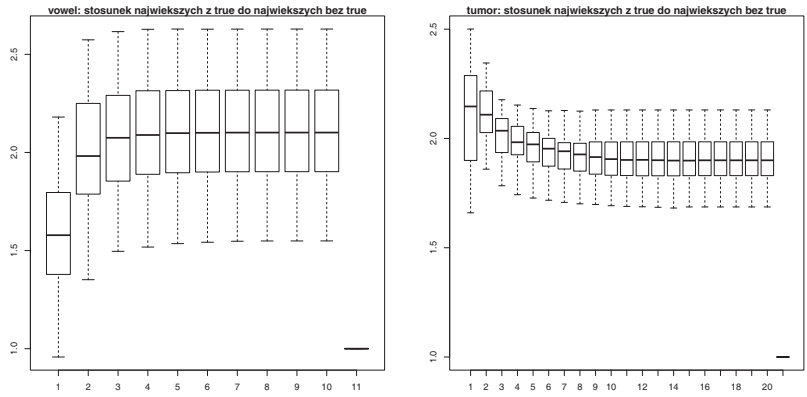
aktywacje niższe. Jeśli zapytamy się o ułamek

$$\left(Cl_{tr}(x) + \sum_{i=1,(i)\neq tr}^{p-1} Cl_{(i)}(x) \right) / \left(\sum_{j=1,j\neq tr}^{p} Cl_{(j)}(x) \right) \tag{2.20}$$

(odpowiadający wyrażeniu (2.19) z teoretycznego procesu, tu jednak dla nauczonej słabo sieci), to otrzymamy wykresy analogiczne pokazane na rysunku 2.14.

Stosunek wag klastrów zawierających prawdziwą klasę do wag klastrów *nie* zawierających prawdziwej klasy zależy od poziomu nauczenia. Przedstawione to jest na rysunku 2.15. Pokazane są tam symulacje dla różnych problemów nauczonych do różnych poziomów średniej aktywacji dla prawdziwej klasy. Na rysunku zaznaczone zostały (linią przerywaną) także poziomy sieci nauczonych do granicznego poziomu słabego klasyfikatora określonego przez wartość $\alpha(K)$ dla K-klasowego problemu. Widać, że dla sieci nauczonych lepiej niż do $\alpha(K)$ już dla dwuelementowych klastrów stosunek wag klastrów jest powyżej 1.

Na rysunkach 2.16 i 2.17 pokazane są (dla różnych zbiorów uczących i poziomów nauczania) wszystkie pary $(w_{\neg tr}(x), w_{tr}(x))$, gdzie $w_{tr}(x)$ to waga klastra (dla przykładu x) zawierającego prawdziwą dla x, a $w_{\neg tr}(x)$ to waga klastra *nie* zawierającego prawdziwej klasy. Przykłady zostały wygenerowane przez nauczenie Cl w korzeniu, klastrowanie wyników jednym z algorytmów, a następnie policzenie, dla wszystkich przykładów, wag dla wszystkich klastrów. Wśród dostępnych klastrów wygenerowane zostały wszystkie pary wag $(w_{\neg tr}(x), w_{tr}(x))$.

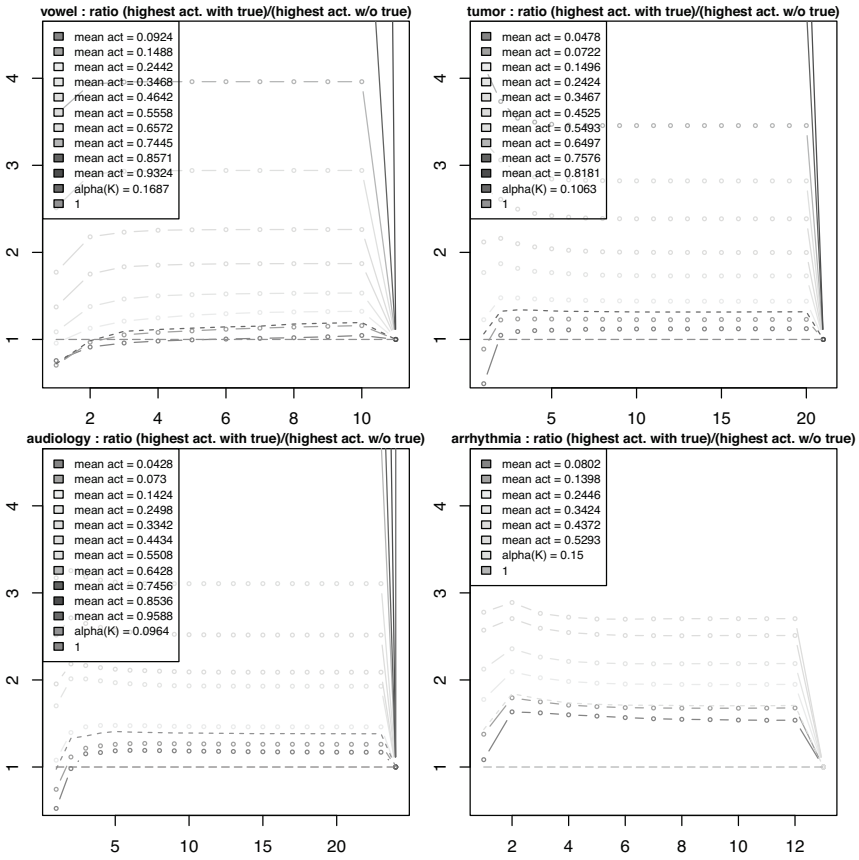


Rysunek 2.14. Stosunek sumy aktywacji dla klasy prawdziwej oraz $p-1$ innych najwyższych aktywacji do sumy p największych aktywacji *bez* aktywacji dla klasy prawdziwej. Uśrednione wyniki dla szeregu doświadczeń dla słabo nauczonych klasyfikatorów dla zbiorów vowel i tumor. Na osi poziomej liczba klas w klastrze

Znaczna większość par znajduje się powyżej osi $y = x$ (dla lepiej nauczonych klasyfikatorów było to zwykle ok. 90–95% wszystkich par). Część par znajduje się poniżej tej osi – dzieje się tak wtedy, gdy najwyższe aktywacje otrzymują klasy rzadko mylone z prawdziwą, a w związku z tym niekoniecznie we wszystkich klastrach razem z prawdziwą. Jednak ten niekorzystny efekt jest niwelowany przez efekt nakładania się klastrów, gdy klasa prawdziwa pojawia się razem w klastrach również z tymi klasami o najwyższych aktywacjach. W związku z tym dodatkowym warunkiem poprawności macierzy klastrowania F może być, aby *każde* dwie klasy znajdowały się razem w co najmniej jednym klastrze (patrz dyskusja w rozdziale 2.2.1).

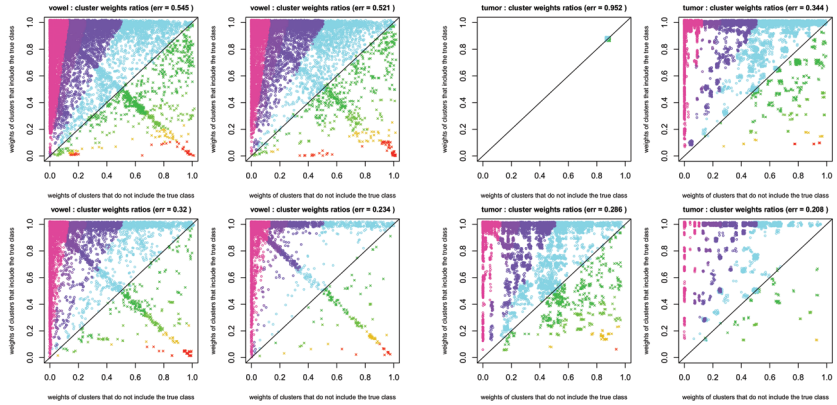
2.4. HCOC jako rozwiązanie zadania przez podział

Klasyfikatory dla problemów wieloklasowych rozwiązują je, biorąc pod uwagę albo wszystkie klasy pod uwagę jednocześnie, albo przez złożenie wielu, zwykle binarnych, klasyfikatorów. Takie podejścia z reguły generują dużą liczbę pojedynczych klasyfikatorów, co może być kosztowne. Casasent, Wang (2005) rozpatrują modyfikację binarnej struktury, w której tworzone są klasyfikatory typu *grupa klas-przeciwko-reszcie*, co pozwala na zmniejszenie liczby klasyfikatorów w stosunku do podejścia *jeden-przeciwko-jednemu* czy *jeden-przeciwko-wszystkim*. Na każdym poziomie klasy są dzielone na dwie równoliczne (z dokładnością do parzystości wyjściowej liczby klas) grupy, wykorzystując specjalnie dostosowaną maszynę typu SVM.

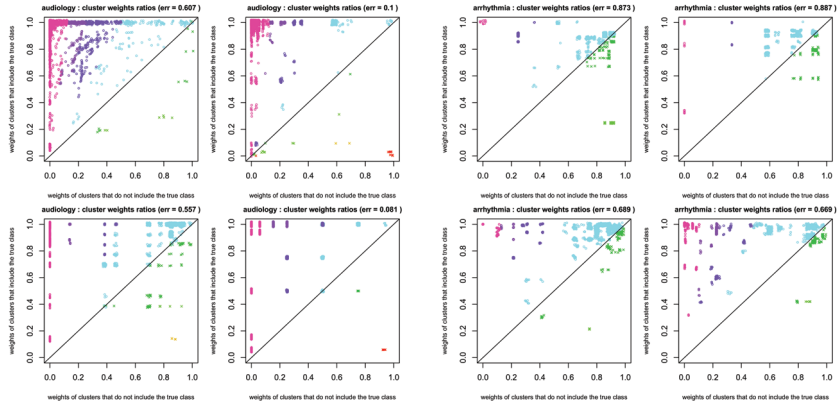


Rysunek 2.15. Stosunek wartości wag klastrów zawierających klasę prawdziwą do równolicznych klastrów *nie* zawierających klasy prawdziwej, ale o najwyższych poziomach aktywacji. Kolejne linie pokazują stosunki dla różnych średnich aktywacji dla klasy prawdziwej. Linią przerywaną zaznaczone średnie dla sieci nauczonych do poziomu $\alpha(K)$. Na osi poziomej wielkość klastrów

Do tego momentu to rozwiązanie przypomina proponowany model HCOC. Jednak w modelu (Casasent, Wang, 2005) na każdym poziomie tworzone są klasyfikatory binarne, klastry nie nakładają się, przetwarzanie jest tylko *w dół*, tzn. klastry są dzielone, aż dojdziemy do odpowiedzi, a nie ma żadnego składania wyników, klasyfikatory w węzłach muszą osiągać jak najlepszy wynik, nie mogą być słabe, aby znaleźć predykcję, trzeba zawsze dojść do liści drzewa, które są zwykle w tej samej odległości od korzenia dla każdej klasy, co zupełnie nie jest założone w HCOC. W rzeczywistości jest to algorytm rekurencyjnego podziału problemu na dwie części, stanowiąc dobre rozwinięcie podejścia typu *jeden-przeciwko-jednemu*.



Rysunek 2.16. Względne wartości wag klastrów zawierających klasę prawdziwą dla przykładu (oś pionowa) do wag wszystkich innych klastrów, które nie zawierają klasy prawdziwej. Wyniki dla zbiorów vowel i tumor. Sieci zostały nauczone aż do osiągnięcia różnych poziomów błędów nauczania. Różne kolory oznaczają od lewej wartości stosunku z zakresów $(\infty, 8)$, $(4, 8]$, $(2, 4]$, $(1, 2]$, $(0.5, 1]$, $(0.25, 0.5]$, $(0.125, 0.25]$, $(0.125, 0.0)$. Dla czytelności, do wektorów dodany został niewielki szum



Rysunek 2.17. Względne wartości wag klastrów zawierających klasę prawdziwą dla przykładu (oś pionowa) do wag wszystkich innych klastrów, które nie zawierają klasy prawdziwej. Wyniki dla zbiorów audiology i arrhythmia

Klasyfikator HCOC proponuje sposób rozdzielenia zadania na podproblemy z pogrupowaniem razem klas, które dzielą z sobą wspólne cechy. Pozwala to na utworzenie mapy problemu, a nie tylko rozwiązanie zadania predykcji. Możliwa

jest np. ekstrakcja reguł typu IF-THEN, stosunkowo dobrze czytelnych dla użytkownika, które dzielą problem na podproblemy rekurencyjnie (Podolak, 2007). Jednocześnie takie podejście może być pewnym rozwiązaniem problemu skalowalności nauczania (Bekkerman *et al.*, 2012).

2.4.1. Klasyfikacja przykładów

Każdy węzeł V^i, oprócz liści, składa się z klasyfikatora Cl^i oraz macierzy klastrującej F^i (która jest w liściach nieobecna). F^i definiują potomne klastry Q^l składające się z podzbioru zbioru klas rozpoznawanych przez V^i (i należących do odpowiadającego mu klastra Q^i). Korzeń oznaczony jest przez V^0. Ponieważ końcowy wynik jest obliczany jako suma ważona aktywacji klasyfikatorów w klastrach potomnych, ważne jest, aby wagi klastrów zawierających klasę prawdziwą dla danego przykładu były wyższe od wag tych, które nie zawierają tej klasy. Jak to zostało pokazane w rozdziale 2.3.4, warunek słabości klasyfikatora w węźle nadrzędnym jest wystarczający.

W trakcie procesu klasyfikacji przez HCOC przykład reprezentowany przez wektor atrybutów $x \in \mathcal{X}$ jest klasyfikowany przez Cl^0 w korzeniu, dając w wyniku wektor aktywacji $Cl^0(x) = [Cl^0_1(x), \ldots, Cl^0_K(x)]$. Ten wektor jest pierwszym przybliżeniem klasyfikacji x.

Uwaga 2.21. *Przestrzeń atrybutów $\mathcal{X}$ może być dla kolejnych klasyfikatorów w węzłach ograniczana. Jeśli jeden z atrybutów nie wpływa na rozwiązanie wydzielonego podproblemu, to może zostać usunięty, polepszając stopień generalizacji. Wymaga to jednak użycia dodatkowego niezależnego algorytmu, co wydłuża czas nauczania (Hand* et al.*, 2001; Hastie* et al.*, 2001; Bishop, 2006; Haykin, 2009).*

Przy okazji pojawia się wiele ciekawych problemów teoretycznych, np. czy jeżeli dany atrybut jest nieistotny dla rozwiązywania danego problemu, to czy jest automatycznie także nieistotny dla wszystkich jego podproblemów? Tematyka redukcji atrybutów wykracza jednak poza ramy tej pracy.

Które z klastrów powinny zostać wybrane? Należy wybrać ten klaster, który maksymalizuje prawdopodobieństwo, że zawiera on prawdziwą klasę przykładu x. Należy wybrać klaster o indeksie l, który maksymalizuje prawdopodobieństwo

$$P(f_{tr(x),l} = 1). \tag{2.21}$$

Zadanie to można zrealizować przez konstrukcję z nauczonego Cl^0 klasyfikatora *uogólnionego* i modyfikację zadania nauczania.

Definicja 2.22. Węzeł $V^i = (Cl^i, F^i)$ nazywamy klasyfikatorem *uogólnionym* (ang. *generalized*), który implementuje funkcję

$$V^i : \mathcal{X} \longrightarrow [0,1]^{L_i}, \tag{2.22}$$

gdzie L_i jest liczbą węzłów pochodnych węzła V^i. Wektor wartości $V^i(x)$ nazywamy wektorem *wag* klastrów, tzn. wartość $V^i_j(x)$ można interpretować jako szansę (ang. *likelihood*), że prawdziwa klasa przykładu x należy do klastra Q^0_j.

Osobne uczenie uogólnionego klasyfikatora mija się z celem, gdyż potrzebna byłaby znajomość postaci klastrów, a tę potrzebujemy właśnie poznać. Należy więc wykorzystać nauczony już klasyfikator Cl^0. Niech $trQ(x; F^0)$ będzie „prawdziwym" wzorcem klastrów dla przykładu x, na tym etapie jeszcze nieznanym:

$$trQ(x; F^0) = [f^0_{tr(x),1}, \ldots, f^0_{tr(x),L^0}], \tag{2.23}$$

który jest binarnym wektorem takim, że $trQ_j(x) = 1$, jeśli $C_{tr(x)} \in Q^0_j$, a w przeciwnym wypadku $trQ_j(x) = 0$. Wektor $trQ(x)$ odpowiada więc prawidłowemu wzorcowi odpowiedzi klasyfikatora uogólnionego przy zadaniu wskazania klastrów zawierających prawidłową klasę przykładu x. Na przykład, jeśli macierz F definiuje podział na klastry $Q_1 = \{C_1, C_2, C_3, C_4\}$, $Q_2 = \{C_3, C_4, C_5, C_6\}$ i $Q_3 = \{C_6, C_7, C_8\}$, a prawidłową klasą danego x jest C_3, to poprawnym wektorem jest $trQ(x) = [1, 1, 0]$, ponieważ C_3 należy do $Q_1 \cap Q_2$.

Uwaga 2.23. *Uogólniony klasyfikator jest klasyfikatorem wieloklasowym*[1] *takim, że dla danego przykładu więcej niż jeden element wyjściowy może być poprawny – dzieje się tak, gdy klasa występuje w więcej niż jednym klastrze. W rzeczywistości prawidłową odpowiedzią jest wskazanie wszystkich takich klastrów. Wskazując jeden klaster uogólniony, klasyfikator podaje w rzeczywistości alternatywę, że prawdziwą klasą jest jedna z należących do tego właśnie klastra.*

Jednocześnie uogólniony klasyfikator podaje alternatywę klastrów. Aby nie była to zwykła alternatywa sumy klas tych klastrów, potrzebne jest ważenie tych odpowiedzi.

Możliwe jest zdefiniowanie odpowiedniej funkcji kosztu opisującej zadanie nauczania uogólnionego klasyfikatora.

Definicja 2.24. Funkcja kosztu opisująca zadanie klasyfikacji przez klasyfikator uogólniony V^0 ma postać

$$\ell(x, trQ(x; F^0), V^0(x)) = 1 - V^0(x) \cdot trQ(x' F^0(x))^T, \tag{2.24}$$

gdzie $trQ(x; F^0)$ opisuje poprawny wzorzec klasyfikacji przykładu x należącego do pewnej ustalonej klasy. Funkcja jest parametryzowana macierzą F^0.

[1] Tu wykorzystywana powszechnie terminologia jest niejednoznaczna: w wielu pracach klasyfikator, który ma możliwe wiele *wykluczających* się odpowiedzi, jest nazywany multiklasyfikatorem W innych pracach to określenie oznacza klasyfikator pozwalający na klasyfikację przykładu do kilku klas jednocześnie. Czasem jest to określenie na złożenie wielu klasyfikatorów. Oczywiście z punktu widzenia samego klasyfikatora pierwsze dwa różnią się jedynie postacią wzorca wyjściowego, jednak klasyfikacja pojedynczego przykładu do wielu klas jednocześnie nie pozwala na przyjęcie założenia o normalizacji wyjścia.

Funkcja kosztu $\ell()$ jest dodatnio określona, a jej wartość jest równa 0 (lub osiąga pewne nieujemne minimum), jeśli wszystkie niezerowe elementy $V^0(x)$ odpowiadają klastrom zawierającym prawdziwą klasę wektora x. Klasyfikator uogólniony wygląda tak jak na rysunku 2.10.

Nauczanie optymalnego klasyfikatora uogólnionego jest procesem znalezienia optymalnej macierzy klastrowania. Proces tworzenia będzie związany z minimalizacją pewnej funkcji dopasowania odpowiadającej funkcji kosztu (2.24).

Definicja 2.25. Niech M będzie macierzą klasyfikacji dla rozpoznającego K klas klasyfikatora Cl. Niech Cl ma L potomków. Niech F będzie macierzą klastrowania o wymiarach $K \times L$. Macierz $G = G(M, F)$ jest $K \times K$ macierzą *generalizacji* zdefiniowaną następująco

$$g_{ii} = \frac{\sum_{k=1}^{K} m_{ik} \sum_{l=1}^{L^0} f_{il}^0 f_{kl}^0}{\sum_{k=1}^{K} m_{ik} \sum_{l=1}^{L^0} f_{kl}^0}, \tag{2.25}$$

$$\text{dla} \quad j \neq i \quad g_{ij} = \frac{\sum_{l=1}^{L^0} m_{ij} f_{jl}^0 (1 - f_{il}^0)}{\sum_{k=1}^{K} m_{ik} \sum_{l=1}^{L^0} f_{kl}^0}. \tag{2.26}$$

Własność 2.26. *Macierz G jest stochastyczna*

Dowód. Pokażemy, że dla ustalonego i zachodzi $\sum_j g_{ij} = 1$:

$$\begin{aligned}
\sum_{j=1}^{K} g_{ij} &= \frac{\sum_l \left[\sum_k m_{ik} f_{il}^0 f_{kl}^0 + \sum_{k \neq i} m_{ik} f_{kl}^0 (1 - f_{il}^0)\right]}{\sum_k \sum_l f_{kl}^0 m_{ik}} \\
&= \frac{\sum_l \left[\sum_k m_{ik} f_{il}^0 f_{kl}^0 + \sum_{k \neq i} m_{ik} f_{kl}^0 - \sum_{k \neq i} m_{ik} f_{il}^0 f_{kl}^0\right]}{\sum_k \sum_l f_{kl}^0 m_{ik}} \\
&= \frac{\sum_l \left(f_{il}^0 f_{il}^0 m_{ii} + \sum_{k \neq i} f_{kl}^0 m_{ik}\right)}{\sum_k \sum_l f_{kl}^0 m_{ik}} = \frac{\sum_l \sum_k f_{kl}^0 m_{ik}}{\sum_l \sum_k f_{kl}^0 m_{ik}} = 1.
\end{aligned} \tag{2.27}$$

□

Elementy g_{ii} na przekątnej odpowiadają szansie, aby przykłady z prawdziwej klasy C_i były prawidłowo zaklasyfikowane. Dzieje się tak w następujących przypadkach, gdy wektory x są:

- zaklasyfikowane przez Cl *poprawnie* jako C_i i wobec tego przypisane poprawnie do klastra zawierającego tę poprawną klasę: w takiej sytuacji w liczniku (2.25) dodana będzie wartość m_{ii} przemnożona przez liczbę klastrów zawierających C_i,
  - w związku z tym klasa powinna występować w maksymalnie wielu klastrach; z drugiej strony, powoduje to wydłużenie nauczania, potrzebny jest więc wybór złotego środka;

- zaklasyfikowane przez Cl *niepoprawnie* jako $C_k \neq C_i$: jeśli istnieje klaster Q zawierający C_i (taki klaster musi istnieć z definicji macierzy F), to licznik (2.25) zostanie zwiększony o wartość m_{ik} przemnożoną przez liczbę klastrów, do których należą *zarówno* C_i, jak i C_k,
  - ta własność pokazuje, że jeśli klasy C_i oraz C_k są często mylone, to powinny być wtedy umieszczane w tym samym klastrze.

Ślad macierzy G odpowiada wartości $(1 - \text{błąd klasyfikatora uogólnionego})$. Jeśli Cl jest losowy dla danego i, to $E[m_{ii}] = 1/K$. Jeśli także oczekiwana liczba klas w klastrze jest równa połowie klastrów $E[\sum_k f_{kl}] = K/2$, a oczekiwana liczba klastrów, do których należy klasa C_i, to $E[\sum_l f_{il}] = L/2$, wtedy oczekiwana wartość elementu na przekątnej macierzy generalizacji to $E[g_{ii}] = 1/2$, co odpowiada losowemu przypisywaniu do klastrów. Ślad nie będzie wysoki, co odpowiada wysokiemu błędowi. Maksymalizacja śladu macierzy G nie jest jedyną miarą dla optymalizacji macierzy klastrowania F.

2.5. Nauczanie pojedynczych węzłów

Każdy klasyfikator w węzłach (klasyfikator bazowy) jest nauczany osobno.

Definicja 2.27. Klasyfikator bazowy w węźle HCOC implementuje funkcję $Cl^i : \mathcal{X} \longrightarrow [0,1]^{K^i}$, zwracającą wektor aktywacji $[Cl^i_1(x), \ldots, Cl^i_{K^i}(x)]$ odpowiadających prawdopodobieństwom, że przykład $x \in \mathcal{X}$ należy do jednej z K^i klas klastra Q^i rozpoznawanego przez Cl^i.

Warunek, żeby każdy klasyfikator w węźle zwracał wektor, który można interpretować jako wektor prawdopodobieństw, jest konieczny ze względu na definicję HCOC. Do budowy takiego klasyfikatora mogą więc być wykorzystane sieci neuronowe, klasyfikator Bayesowski, inne modele, które albo zwracają wektor prawdopodobieństw wprost, albo mogą być tak zmodyfikowane, aby to robiły.

Bardzo istotny jest sposób nauczania. Podstawowym założeniem konstrukcji HCOC jest to, że algorytm nauczania wykorzystuje *niedokładności* nauczenia klasyfikatora w węźle Cl^i, żeby na ich podstawie wydedukować informacje o rozkładzie gęstości klas i utworzyć z tego informacje o rozkładzie gęstości klastrów. Nie chodzi jedynie o to, aby klasyfikator w węźle podawał maksymalnie często odpowiedź prawidłową, a poza tym popełniał losowe błędy przypisania, ale raczej dawał poprawne odpowiedzi tylko trochę częściej niż przypadkowo (*słabość* klasyfikatora), a poza tym popełniał błędy *systematyczne* (patrz rozdział 2.3.2). Błędy systematyczne można w trakcie budowy, w procesie klastrowania, wyszukać i zniwelować.

Jako klasyfikatory bazowe można wykorzystać różne modele klasyfikacji. Częstym wyborem są sieci neuronowe, które zwracają wektor prawdopodobieństw przynależności do klas konieczny dla działania HCOC. Dobrym rozwiązaniem

może być wykorzystanie prostych modeli nauczania niekoniecznie dających wysoką skuteczność, ale prostych i szybkich w budowie. Może to być las drzew decyzyjnych zbudowany tak, aby uśredniał klasyfikacje poszczególnych drzew i zwracał także wektor prawdopodobieństw.

Las drzew decyzyjnych jako klasyfikator bazowy Klasyfikatory takie jak drzewa decyzyjne zwracają dla każdego przykładu nie wektor prawdopodobieństw, ale indeks wybranej klasy. Nie mogą być więc wprost użyte jako klasyfikatory bazowe HCOC. Jeśli jednak zbudować las drzew, a odpowiedzi uśrednić, to spełniony byłby warunek definicji. Analogicznie algorytmy takie jak SVM podają indeks klasy i potrzebna byłaby analogiczna procedura (chociaż bardziej złożona ze względu na dwuklasowość SVM). Wybór takiego algorytmu może się jednak mijać z celem ze względu na złożoność ich nauczania oraz fakt, że HCOC ma właśnie wykorzystywać cechy słabości klasyfikatorów w węzłach.

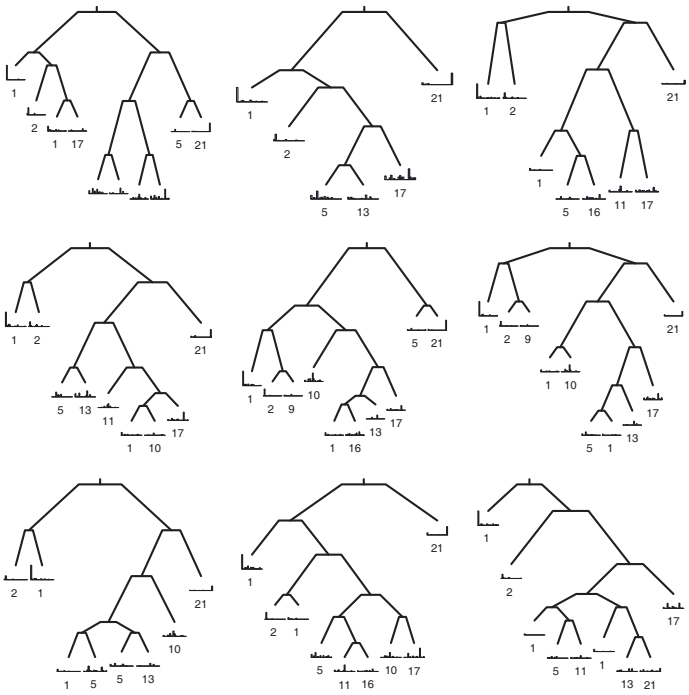
Klasyfikator wykorzystujący drzewa decyzyjne jest zbudowany jako las drzew decyzyjnych. Dla danego N-elementowego zbioru uczącego stosowana jest procedura *bootstrap* (Efron, Tibshirani, 1997) polegająca na losowaniu ze zwracaniem N-elementowych zbiorów uczących. Taka procedura powoduje, że w danym zbiorze *bootstrap* ok. $0.632N$ przykładów, wylosowanych z N-elementowego zbioru uczącego, jest różnych. Przy zastosowaniu wstępnie procedury stratyfikacji mamy pewność, że każdy taki zbiór zawiera przykłady z wszystkich klas (patrz A.13).

Dla każdego zbioru *bootstrap*, wykorzystując algorytm drzew decyzyjnych, budowane jest osobne drzewo, przy czym każde drzewo jest głęboko obcinane (ang. *pruning*), aby uzyskać liście składające się z wielu klas. W całym podejściu budowany jest cały las drzew, złożony z 50–100 niezależnych klasyfikatorów. W trakcie predykcji drzewo decyzyjne standardowo zwraca etykietę większościową liścia, w którym kończy się klasyfikacja. Zamiast tej etykiety można wziąć rozkład prawdopodobieństw klas wszystkich przykładów ze zbioru uczącego, które w trakcie nauczania zostały umieszczone w liściu wygrywającym. Na rysunku 2.18 pokazanych jest dziewięć drzew zbudowanych dla problemu *breast tumour* (Newman *et al.*, 1998). W trakcie klasyfikacji danego wektora atrybutów x ewaluowane są wszystkie nauczone drzewa. Dla klas, których przykłady nie są zawarte w zwycięskim liściu, zwracana jest wartość 0. Wektory klas prawdopodobieństwa wszystkich liści $Cl_t(x)$ są uśredniane, dając końcową klasyfikację

$$Cl_F(x) = \frac{1}{N} \sum_t Cl_t(x). \tag{2.28}$$

Dzięki takiemu podejściu

- sterując liczbą użytych drzew, można sterować poziomem nauczenia takiego lasu drzew,
- wykorzystanie lasu drzew daje lepszą generalizację niż wykorzystanie pojedynczego drzewa, dzięki czemu wartość wektora aktywacji $Cl_F(x)$ będzie



Rysunek 2.18. Pojedyncze drzewa decyzyjne dla problemu *breast tumour* (Newman *et al.*, 1998). Każde zbudowane na podstawie osobnego zbioru *bootstrap*

dawać dobre przybliżenie wartości wag klastrów, a model łączenia wyników oparty na danych, jakim jest HCOC, będzie miał uzasadnienie,

- ponieważ nauczanie pojedynczego drzewa jest szybkie, można w prosty sposób zbudować klasyfikator w węźle, który będzie rozdzielać problem złożony z bardzo wielu klas na mniej liczne grupy, gdzie można zastosować klasyfikatory dające lepszy stopień generalizacji, np. sieci neuronowe,
- można wykonać podział na klastry już w trakcie nauczania: każdy liść definiuje klaster jako mapę klas do niego należących; to alternatywny sposób klastrowania względem opisanych w rozdziale 2.6 metod.

2.6. Zadanie klastrowania

Zadanie klastrowania polega na znalezieniu optymalnej macierzy F takiej, że zminimalizowana jest pewna funkcja kosztu, np. określona w definicji 2.24. Możliwe jest wiele podejść:

- rozszerzające hierarchiczne klastrowanie aglomeratywne (Day, Edelsbrunner, 1984) dla znalezienia klastrów nakładających się; algorytm wykorzystuje podobieństwo wierszy macierzy M (Podolak, 2008; Podolak *et al.*, 2006);

- oparte na prawie Bayesa wykorzystuje macierz pomyłek $M = (m_{ij})_{i,j=1}^{K}$ taką, że $m_{ij} \simeq P(pr = C_j|tr = C_i)$, ewaluowaną w trakcie nauczania klasyfikatora Cl^i w korzeniu aktualnego poddrzewa HCOC; wykorzystując prawo Bayesa, znajdujemy $P(tr = C_j|pr = C_i)$ i, rozpoczynając od diagonalnej macierzy F opisującej jednoelementowe klastry, dodajemy kolejne o najwyższych $P(tr = C_j|pr = C_i)$, dbając o poprawność F (Podolak, Roman, 2011a);

- wykorzystanie adaptacyjnych metod nauczania maszynowego przez użycie algorytmu Rosnącego Gazu Neuronowego (Fritzke, 1995) w przestrzeni wyjsciowej; ten algorytm nadaje się do realizacji zadania klastrowania *online* przez urównoleglenie klastrowania z nauczaniem samego klasyfikatora; pozwala to na przerwanie nauczania w momencie znalezienia optymalnego podziału (Podolak, Bartocha, 2009);

- wykorzystanie algorytmów genetycznych ze zdefiniowaną funkcją dopasowania, co ma tę zaletę, że można łatwo uwzględnić w funkcji dopasowania inne cele, jak np. zapewnienie różnorodności klasyfikatorów pochodnych (Podolak, Roman, 2011a);

- przeglądanie wszystkich macierzy F – niemożliwe ze względu na olbrzymią liczbę i trudność w ich wyszukania (patrz rozdział 2.2.1) (Podolak *et al.*, 2012b).

2.6.1. Rozszerzenie algorytmu klastrowania aglomeratywnego

Podstawowe podejście do podziału na podproblemy polega na rozszerzeniu klastrowania przez łączenie (ang. *agglomerative*) SAHN (*Simple Agglomerative Hierarchical Clustering*) do zadania, w którym klastry mają się nakładać (Day, Edelsbrunner, 1984; Podolak, 2008). SAHN jest z założenia algorytmem tworzącym klastry *rozłączne*, stąd ten algorytm musiał zostać zmodyfikowany, aby dobrze odpowiadać zadaniu dla HCOC[2].

Metody aglomeratywne rozpoczynają rozwiązanie zadania od n jednoelementowych klastrów. Kolejne (większe) klastry są tworzone zwykle przez łączenie par dotąd utworzonych klastrów. Wybór polega zwykle na wstępnym utworzeniu macierzy odległości $D = (d(x, y))_{x,y\in\mathcal{X}}$ między wszystkimi parami obiektów, a następnie łączeniu tych o najmniejszych odległościach. Sama miara $d()$ zależeć będzie od istoty problemu. Przy klastrach zawierających więcej niż jeden element

[2]Niektórzy autorzy używają akronimu SAHN specyficznie dla określenia grupy metod klastrowania, które są sekwencyjne, aglomeratywne (kolejne klastry są tworzone przez łączenie), hierarchiczne i generują klastry nienakładające się (Sneath, Sokal, 1973).

możliwych jest kilka strategii: najbliższego sąsiada, najdalszego sąsiada, średniej grupowej (patrz dodatek A.8).

W większości przedstawionych dalej rozwiązań klastrowania dla problemu HCOC przyjęte zostało podejście uśredniające łączące klastry o najbliższych centroidach. W ten sposób obiektami w pojedynczym klastrze są klasy, które dany klasyfikator często z sobą *wzajemnie* myli[3]. Klastry powinny być raczej zwarte, niektóre rozwiązania mogą czasem iść w kierunku strategii najbliższego sąsiada: np. podejście Bayesowskie dodaje kolejno klasy najczęściej mylone z klasami już obecnymi w klastrze (patrz opis w rozdziale 2.6.2). Podejście typu GNG (patrz rozdział 2.6.3) wykorzystuje podobieństwo wszystkich klas do siebie, jest więc, w większym stopniu, wyborem strategii średniej grupowej. Zwartość klastrów nie zawsze jest do osiągnięcia, a metoda najdalszego sąsiada może czasem dawać różne wyniki przy wahaniach poziomu i drobnych modyfikacjach zbioru uczącego.

Niech Cl będzie klasyfikatorem bazowym w korzeniu poddrzewa budowanego klasyfikatora HCOC. Cl jest nauczony co najmniej do poziomu bycia słabym. Dla każdego przykładu ze zbioru uczącego reprezentowanego przez wektor atrybutów x, Cl zwraca K-wymiarowy wektor aktywacji. Obiektami, które mają być klastrowane, są średnie wektory aktywacji dla przykładów z każdej ze zdefiniowanych klas z osobna. Dla danej klasy C_i średnia aktywacja odpowiada wierszowi macierzy pomyłek $m_{i*} = [m_{i1}, \ldots, m_{iK}]$. Cały proces klastrowania oparty jest na macierzy M, nie na wektorach aktywacji dla poszczególnych przykładów uczących (tak dzieje się w, opisanym dalej, podejściu wykorzystującym algorytm GNG).

Celem jest umieszczenie w jednym klastrze klas często z sobą mylonych. Algorytm iteruje więc po wierszach macierzy M odpowiadających prawdziwym klasom. Dla prawdziwej klasy C_i (wiersza) znajduje inną klasę C_j (odpowiadającej kolumnie) o wysokiej wartości m_{ij}, tworząc początkowe dwuelementowe klastry. W następnych krokach wybiera i dodaje do już znalezionych klastrów inne klasy mylone z prawdziwą klasą C_i. Według użytej heurezy wyszukiwane jest $\lfloor K/4 \rfloor$ do $\lfloor K/2 \rfloor$ klas poza prawdziwą. Klastry, które są podzbiorami innych, są zwykle usuwane. Analogicznie klastry o rozmiarze mniejszym od minimalnego są także usuwane. Algorytm opisany jest na rysunku 2.19.

Każdy klaster Q^i definiujemy jako

$$Q^i = \{C_i, C_{(1)}, C_{(2)}, \ldots, C_{(\lfloor K/4 \rfloor)}\}, \tag{2.29}$$

gdzie $C_{(j)}, (j) \neq i$ są klasami odpowiadającymi wartościom wiersza m_{i*} posortowanym w kolejności malejącej, a więc klasom o najwyższej wartości $P(C_j|tr(x) = C_i)$ prawdopodobieństwa niepoprawnego rozpoznania x z klasy C_i jako obiektu klasy C_j.

[3]To także zależy od postaci konkretnej funkcji kosztu minimalizowanej w procesie.

Input: M – macierz pomyłek
$groups \leftarrow \emptyset$ // mapowanie klasyfikacji przez Cl do prawdziwych klas
for *column in* $1 : K$ **do**
  // iteracja po kolumnach macierzy M
  $classAs \leftarrow \emptyset$ // elementy indeksowane prawdziwa klasa
  $thisColumn \leftarrow \{(M[row, column] * P(C_{row}), row)\}_{row=1}^{K}$
  $classAs \leftarrow selectHighestValues(bc, \lfloor K/4 \rfloor)$
  $classAs \leftarrow classAs \cup column$
  if $groups == \emptyset$ **then**
    $groups \leftarrow groups \cup (column, classAs)$
  else
    // scal klastry zawarte w innych klastrach
    for $(trueClass, classified) \in groups$ **do**
      if $classAs \subseteq classified$ **then**
        $groups \leftarrow groups \setminus (true, classified)$
        $trueClass \leftarrow trueClass \cup column$
        $groups \leftarrow groups \cup (trueClass, classified)$
      else if $classified \subseteq classAs$ **then**
        $groups \leftarrow groups \setminus (trueClass, classified)$
        $trueClass \leftarrow trueClass \cup column$
        $groups \leftarrow groups \cup (trueClass, classAs)$
      end
      $groups \leftarrow groups \cup (column, classAs)$
    end
  end
end
// sklej razem niewielkie klastry
while $(ha, a, hb, b) = MinSum(groups)$ *and* $size(a \cup b) < minSize$ **do**
  $groups \leftarrow groups \setminus (ha, a)$; $groups \leftarrow groups \setminus (hb, b)$
  $groups \leftarrow groups \cup (ha \cup hb, a \cup b)$
end
return $groups$

Rysunek 2.19. Algorytm klastrowania poprzez rozszerzenie algorytmu SAHN. Operator $\setminus$ oznacza odejmowanie zbiorów

2.6.2. Bayesowskie podejście do klastrowania

Elementy m_{ij} macierzy M ewaluują z wyników nauczania prawdopodobieństwo warunkowe $P(C_j|C_i, Cl)$ predykcji przez Cl przykładu z prawdziwej klasy C_i jako C_j. Zgodnie z założeniem, jeśli $C_{tr(x)}$ jest prawdziwą klasą dla x, to klasy $C_{(y)}$, dla których wartość $P(C_{(y)}|C_{tr(x)}, Cl)$ jest wyższa niż dla innych klas, powinny

znaleźć się w tym samym klastrze co $C_{tr(x)}$. Algorytm Bayesowski generuje klastrowanie według następującego schematu

1. ewaluuj $\widehat{P}(C_j|C_i) = m_{ij}$,
2. utwórz K początkowych rozłącznych klastrów jako singletony składające się z jednej klasy każdy,
3. posortuj wartości $\widehat{P}(C_j|C_i)$ w kolejności malejącej,
4. oblicz macierze klastrowania F, generalizacji G (patrz definicja 2.25) i ślad $tr(G)$,
5. powtarzaj
   (a) weź najwyższą wartość $\widehat{P}(C_j|C_i)$ i dodaj klasę przewidywaną C_j do klastra zbudowanego wokół klasy C_i,
   (b) oblicz tymczasowe F', usuwając klastry zawarte w innych klastrach,
   (c) oblicz nowe G i akceptuj zmiany tylko wtedy, gdy modyfikacja zwiększa $tr(G)$.

Dokładny algorytm podany jest na rysunku 2.20.

Wynik działania procedury zależy od kolejności, w której wstawiane są nowe klasy. Może się zdarzyć, że różnice między kolejnymi elementami m_{ij} są niewielkie, a ich kolejność po posortowaniu będzie różna przy niewielkiej modyfikacji zbioru uczącego, np. przykłady z jednej z klas będą częściej reprezentowane. Z tego wynikają 2 warianty podstawowej procedury:

- w podstawowym trybie element m_{ij} może zostać pominięty, ponieważ jego dodanie nie spowodowało wzrostu $tr(G)$; pierwsza modyfikacja (tzw. *niezależna od kolejności*) będzie polegała na powtórnej próbie użycia odrzuconych wcześniej elementów m_{ij};
- może się zdarzyć sytuacja, że po dodaniu klasy C_j odpowiadającej najpierw m_{ij}, a bezpośrednio potem klasy C_q odpowiadającej m_{pq} (jedno i drugie spowodowało wzrost $tr(G)$), *usunięcie* teraz klasy C_j spowoduje dalszy wzrost $tr(G)$; jest to związane z zależnościami pomiędzy sobą rozpoznawania klas C_j i C_q oraz niewielkimi różnicami w wartościach m_{ij} oraz m_{pq}; druga modyfikacja (tzw. *usuwająca pary*) będzie polegała właśnie na sprawdzeniu możliwości usunięcia pierwszej klasy po każdych dwóch dodanych; w efekcie dostajemy klastry, które są bardziej zwarte.

Kosztem niewielkiego wzrostu złożoności obliczeniowej obydwa podejścia powodują uzyskanie bardziej równolicznych klastrów. Wagi klastrów są proporcjonalne do sumy aktywacji klasyfikatora nadrzędnego dla klas zawartych w danym klastrze (patrz wzór (2.10)). Jeśli klastry będą bardziej równoliczne, zmniejszy się prawdopodobieństwo, że waga klastra *nie* zawierającego klasy prawidłowej przeważy nad wagą klastra, który tę klasę zawiera.

Input: M – macierz pomyłek
Input: $maxClusterSize$ – maksymalny rozmiar klastra
Output: F – macierz klastrowania
$M' = computeBayes(M)$ // obliczenie $\hat{P}(true = C_j | predicted = C_i)$
$F \leftarrow I^{K \times K}$ // start z singletonami
$gtr \leftarrow \sum_i m'_{ii}$
$mv \leftarrow order(M')$ // indeksowanie M malejąco
for $l \leftarrow 1$ *to* $length(mv)$ **do**
  $(i, j) \leftarrow mv[l]$ // kolejna para klas (i, j)
  $F' \leftarrow F$ // kopia robocza
  $mx \leftarrow \sum_l f'_{il}$
  if $i \neq j$ || mx <= $maxClusterSize$ **then**
    $f'_{ij} \leftarrow 1$ // usuń klastry zawarte w innych
    $F'' \leftarrow reduceClusteringMatrix(F')$
    $G \leftarrow generalizationMatrix(M, F'')$ // macierz generalizacji G
    $gtr' \leftarrow trace(G)$
    // add (i, j) tylko, gdy $tr(G)$ wzrasta
    if $gtr' > gtr$ **then**
      $F \leftarrow F'$
      $mv[l].added \leftarrow TRUE$
    else
      $mv[l].added \leftarrow FALSE$
    end
    if *addSkipped* **then** $(F, mv, gtr) \leftarrow addSkipped(F, mv, gtr)$
    if *removeUnneeded* **then** $(F, mv, gtr) \leftarrow removeUnneeded(F, mv, gtr)$
  end
end
$F \leftarrow reduceClusteringMatrix(F)$ // ostateczna macierz F
return F

Rysunek 2.20. Algorytm klastrowania Bayesowskiego

2.6.3. Urównoleglenie klastrowania przy wykorzystaniu algorytmu Rosnącego Gazu Neuronowego GNG

Wydaje się, że urównoleglenie procesów nauczania klasyfikatora bazowego *Cl* w węźle oraz klastrowania dla niego wyników powinno przynieść szczególnie dobre wyniki. Jest to istotne nie tylko ze względu na możliwe przyspieszenie obliczeń, ale przede wszystkim dlatego, że:

- wynik klastrowania jest w dużym stopniu zależny od poziomu nauczenia klasyfikatora bazowego *Cl*. Niech *Cl* będzie bardzo mało dokładnym (bar-

dzo słabym) klasyfikatorem, który jednak zwraca wyniki z dużą pewnością, tzn. dla zwycięskiej klasy C_j daje aktywacje Cl_j bliskie 1, bez względu na to, czy C_j jest klasą prawdziwą, czy nie. W takiej sytuacji dla niektórych klas tworzą się bardzo małe klastry i efekt klastrowania nie zmniejsza błędu HCOC w wystarczającym stopniu. Bierze się to z faktu istnienia minimów lokalnych. Końcowy HCOC będzie sobie często radził lepiej dzięki wcześniejszemu zakończeniu nauczania *Cl* i skorzystaniu z efektu klastrowania. Można to zapewnić przez jednoczesne uruchomienie nauczania i klastrowania oraz przerwanie obu procesów w najbardziej dogodnym momencie;

- konieczne jest zapewnienie różnorodności klasyfikatorów bazowych: jeśli wszystkie klasyfikatory pochodne będą popełniały te same, lub prawie te same błędy, to nie należy oczekiwać dużego wzrostu poprawności dzięki samemu składaniu wyników (dokładniejsza dyskusja w rozdziale 2.6.7). Jest to możliwe do osiągnięcia przez kontrolę obydwu procesów.

Do urównoleglenia nauczania i klastrowania można wykorzystać algorytm rosnącego gazu neuronowego (ang. *Growing Neural Gas*, GNG) (Fritzke, 1995). Algorytm GNG tworzy graf węzłów (neuronów), których liczba, a także połączenia pomiędzy nimi zmieniają się w trakcie nauczania. Każdy węzeł jest związany z wektorem wag $w_i \in \mathbb{R}^K$ (bardziej szczegółowy opis algorytmu GNG znajduje się w dodatku A.9). W przypadku klastrowania dla HCOC będą to wektory aktywacji (pojedynczych lub małych grup bardzo podobnych) dla przykładów uczących.

GNG tworzy, w procesie konkurencyjnego nauczania nienadzorowanego, klastry dla aktywacji klasyfikatora. Te klastry odpowiadają podobnym aktywacjom. Jeśli więc aktywacje dla przykładów z różnych prawdziwych klas są łączone w ten sam klaster, może to oznaczać, że przykłady dla tych klas są z sobą mylone. Będzie to pociągało za sobą łączenie ich w klastry HCOC. Zaletą użycia GNG jest fakt, że algorytm bardzo dobrze dostosowuje się do zmieniającego się środowiska (Fritzke, 1997). Możliwe jest więc rozpoczęcie klastrowania na podstawie wstępnie nauczonego *Cl*, a następnie kontynuowanie nauczania. Procesy nauczania i klastrowania są przerywane po osiągnięciu ustalonego celu nauczania.

Ważną różnicą względem wcześniejszych algorytmów jest fakt, że przy użyciu GNG klastrowane będą klasy na podstawie wektorów aktywacji dla pojedynczych przykładów, nie ich uśrednionych aktywacji znalezionych po zakończeniu nauczania. Procesy nauczania i klastrowania mogą być wykonywane równolegle.

Najistotniejszym elementem definicji wykorzystania GNG dla klasyfikatora HCOC jest zdefiniowanie sposobu związania z każdym utworzonym klastrem jego opisu w postaci listy zebranych tam klas. Celem klastrowania HCOC jest utworzenie nakładających się grup klas często z sobą mylonych. Po utworzeniu klastrów GNG potrzebna jest analiza, dla których prawdziwych klas wektory aktywacji są połączone w jednym klastrze. Bez odpowiedniej analizy często w utworzonych klastrach wystąpią wektory aktywacji dla wszystkich lub prawie wszystkich klas, co jest oczywiście wynikiem nieprawidłowym.

Sam etap klastrowania GNG łączy z sobą przykłady, które mają podobne aktywacje, HCOC potrzebuje jednak klastrów klas. Aby to uzyskać, dla każdego GNG klastra G tworzony jest histogram $H : Q^0 \longrightarrow [0,1]$, gdzie Q^0 jest klastrem zawierającym wszystkie klasy, a $H(C_i)$ reprezentuje frakcję, z jaką klasa C_i jest prawdziwą klasą przykładów przypisanych do tego klastra. Każdy GNG-klaster odpowiada HCOC klastrowi Q^G takiemu, że

$$Q^G = \{C_i : H(C_i) > d_G\}, \tag{2.30}$$

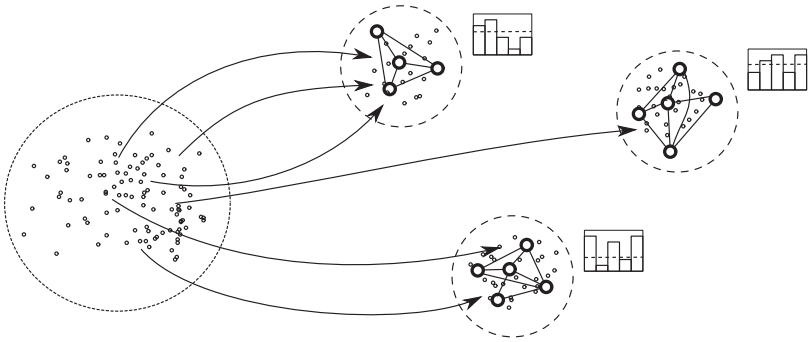
gdzie d_G jest pewnym progiem. Algorytm znajduje takie d_G, aby założenia poprawności macierzy klastrowania F (patrz definicja 2.11) były spełnione. Jeśli istnieje taka klasa C_j, że C_j *nie* należy do żadnego znalezionego w ten sposób klastra, to należy wybrać klaster maksymalizujący frakcję dla C_j:

$$G = \arg\max_{G'} H_{G'}(C_j), \tag{2.31}$$

$$G = G \cup C_j. \tag{2.32}$$

Najprostszym wyborem d_G jest początkowe ustalenie $d_G = \frac{2}{3}\max_i H_G(C_i)$, a następnie proste dopasowanie. Działanie GNG jest pokazane na rysunku 2.21.

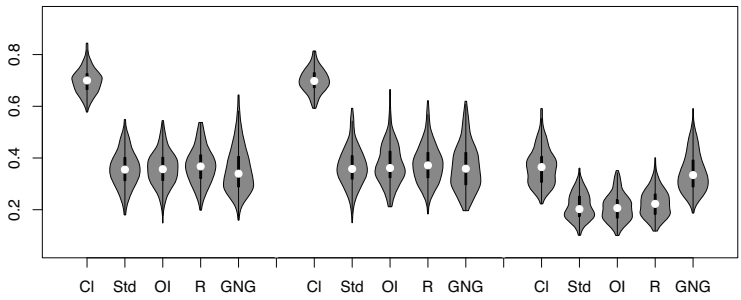
Tak jak wcześniej opisane algorytmy typu SAHN i Bayesowskie są oparte na bezpośrednim wykorzystaniu prawa Bayesa, tak rozwiązanie GNG łączy w grupy klasy, dla których wektory aktywacji są podobne w sensie użytej miary odległości (zwykle euklidesowej). Tworzone klastry są wobec tego bardziej zwarte, a włączone klasy mylone każda z każdą, bardziej niż to się dzieje w przypadku klastrowania Bayesowskiego.



Rysunek 2.21. Klastrowanie z wykorzystaniem GNG. Od lewej do prawej: przestrzeń atrybutów, odwzorowanie przez Cl do przestrzeni aktywacji podzielonej przez GNG na klastry, histogramy częstotliwości klas w poszczególnych klastrach GNG, wybór progów dla ustanowienia klastrów Q. Na pokazanym przykładzie utworzone są klastry złożone z klas $\{C_1, C_2\}$, $\{C_2, C_3, C_5\}$ oraz $\{C_1, C_3, C_4\}$

Na rysunku 2.22 pokazane jest porównanie wyników klastrowania typu Bayesowskiego i typu GNG. Wykresy pokazują średnie błędy i ich rozrzut dla HCOC klastrowanych metodą Bayesowską (podejścia standardowe STD, niezależne od kolejności OI (ang. *Order Invariant*) i z usuwaniem par R (ang. *Removal*)) oraz metodą GNG. Wyniki podane są dla różnego stopnia nauczenia klasyfikatorów bazowych (od lewej na rysunku): do poziomu błędu $1-\alpha(K)$, a więc prawie losowe, do $1-2\alpha(K)$ oraz $1-4\alpha(K)$. Wyniki pochodzą z doświadczenia dla problemu vowel. W każdej grupie nauczonych zostało 200 trzypoziomowych sieci. Dla porównania po lewej stronie, z indeksem CL podany jest wynik osiągany przez klasyfikator w korzeniu.

Wyniki pokazują, że dla słabiej nauczonych klasyfikatorów bazowych wyniki działania algorytmów Bayesowskiego i GNG są porównywalne. Dla lepiej nauczonych zdecydowanie lepsze wyniki daje podejście Bayesowskie.



Rysunek 2.22. Porównanie metod klastrowania Bayesowskiego i GNG. Wykresy typu wiolinowego pokazują średnie błędy i ich rozrzut dla problemu vowel. Wszystkie sieci ewaluowane były metodą RESTRICTED

2.6.4. Wykorzystanie metod genetycznych dla klastrowania

Innym podejściem może być wykorzystanie algorytmu typu genetycznego (Holland, 1992; Michalewicz, 1996; Schaefer, Telega, 2007). W klastrowaniu dla HCOC optymalizacji poddane zostało zadanie znalezienia najlepszej macierzy klastrowania F przy wykorzystaniu operacji krzyżowania macierzy.

Definicja 2.28. Niech F_1 i F_2 będą macierzami klastrowania o wymiarach $K \times L_1$ i $K \times L_2$. Niech $b_1 \in \{1,\ldots,L_1-1\}$ i $b_2 \in \{1,\ldots,L_2-1\}$ będą punktami podziału kolumn F_1 i F_2. W wyniku operacji krzyżowania nowa macierz F_1' powstaje jako złożenie kolumn $1,\ldots,b_1$ z F_1 i kolumn $(b_2+1),\ldots,L_2$ macierzy F_2, analogicznie F_2' jako kolumny $1,\ldots,b_2$ macierzy F_2 i $(b_1+1),\ldots,L_1$ macierzy F_1. Wyjściowe F_1'' i F_2'' powstają z F_1' i F_2' przez usunięcie kolumn zawartych w innych oraz dodanie klas tak, aby spełnione były warunki poprawności klastrowania z definicji 2.11.

Definicja 2.29. Operacja mutacji macierzy klastrowania F polega na wyborze pary indeksów (k,l) i podmianie $f_{kl} = 1 - f_{kl}$. Dodatkowo mutacja polega na losowym dodawaniu lub usuwaniu całych klastrów odpowiadającym kolumnom F. Ewentualnie konieczna może być taka modyfikacja F, aby spełnione były warunki definicji 2.11.

Schemat algorytmu pokazany jest na rysunku 2.23. W klastrowaniu genetycznym dla HCOC, pokazanym w wynikach w tabeli 2.2, wykorzystywane były początkowe populacje z 50 macierzami. Przez kilkadziesiąt iteracji tworzone były nowe populacje dla optymalizacji założonej funkcji dopasowania (*fitness*). Funkcje dopasowania wykorzystywały informacje z macierzy pomyłek M, macierzy generalizacji G, postaci klasyfikatora Cl w aktualnym korzeniu, postaci aktualnego modelu HCOC, obliczanej funkcji różnorodności, także zgodności F z warunkami jej poprawności wynikającymi z definicji. Macierz generalizacji G jest funkcją M i aktualnej macierzy klastrowania F: $G = G(M,F)$, jest więc obliczana osobno dla każdej ocenianej macierzy w populacji.

Wyniki dla algorytmów Bayesowskiego i genetycznego są porównywalne, jednak Bayesowskie są dużo szybsze, stąd są rozwiązaniem preferowanym.

2.6.5. Inne funkcje dopasowania

Maksymalizacja śladu macierzy G nie jest jedynym podejściem do rozwiązania problemu klastrowania. Można wykorzystać szereg innych wskazówek dla dobrego klastrowania, pochodzących z rozważań teoretycznych dotyczących HCOC.

Wykorzystanie górnej granicy ryzyka

Podejście możliwe jest dzięki ewaluacji macierzy pomyłek $\widetilde{M}^l$ (patrz rozdział 1, wzór (1.18)) klasyfikatora pochodnego Cl^l, obliczone na podstawie działania aktualnego klasyfikatora jako

$$\widetilde{m}_{ik} = m_{ik} + \sum_{j \notin \mathcal{I}} m_{ij} \frac{P(tr = C_k | pr = j)}{\sum_{l \in \mathcal{I}} P(tr = l | pr = j)}. \tag{2.33}$$

Jednocześnie wykorzystując ograniczenie funkcji kosztu dla HCOC (po podstawieniu $\widetilde{M}^l$ za M^l) jako (patrz równanie (2.49)):

$$E[\ell(HCOC)] = 2 - \frac{2}{B} \sum_{i=1}^{K} \sum_{l=1}^{L^0} \sum_{k=1}^{K} f_{kl}^0 \widetilde{m}_{ii}^l m_{ik}^0, \tag{2.34}$$

gdzie B jest zależnym od M i F czynnikiem normalizującym. Tę miarę można wykorzystać jako bezpośrednią funkcję dopasowania macierzy F, korzystając z ewaluacji macierzy pomyłek $\widetilde{M}^l$ klasyfikatorów potomnych.

Input: M macierz pomyłek Cl
Input: $HCOC$ aktualny klasyfikator HCOC
Output: F wynikowa macierz klastrowania
// początkowa populacja macierzy klastrowania
$Pop(F) \leftarrow inicjalnaPopulacjaMacierzyKlastrowania(M, Cl, HCOC, n)$
// wartości dopasowania macierzy F
$fit \leftarrow fitness(Pop, M, Cl, HCOC)$
while $not(stopCondition)$ **do**
  // nowa populacja wraz z wartością dopasowania każdej
  $NewPop(F) \leftarrow nowaPopulacja(Pop, fit)$
  // nowa populacja przez krzyżowanie najlepszych macierzy klastrowania
  $NewPop(F) \leftarrow crossover(NewPop)$
  // operator mutacji na macierzach klastrowania
  $Pop \leftarrow mutacja(NewPop)$
  // wartości dopasowania macierzy klastrowania w nowej populacji
  // obliczone na podstawie M, Cl, $HCOC$
  $fit \leftarrow fitness(Pop, M, Cl, HCOC)$
  // przechowywanie tymczasowej najlepszej macierzy
  $tmpBestF \leftarrow najlepszaMacierz(Pop, tmpBestF)$
end
$F \leftarrow najlepszaMacierz(Pop, tmpBestF)$
return F

Rysunek 2.23. Schemat procedury genetycznej. Warunek końca może być określony przez koniec spadku kosztu, maksymalną liczbę epok lub ustaloną wystarczającą wartość funkcji *fitness*

Kontrola poprawności macierzy F

Ta kontrola jest szczególnie potrzebna przy wykorzystaniu algorytmów genetycznych. W zależności od konkretnej implementacji możliwe jest generowanie macierzy F niespełniających perfekcyjnie wszystkich warunków poprawności, co często ułatwia i przyspiesza zbieżność. Dodanie prostych warunków kontrolujących liczbę klastrów, uwzględnienie wszystkich klas, przecinanie się klastrów itp. także ułatwia zbieżność. Możliwe jest dodanie innych warunków heurystycznych, np. przewidujących, ile klas powinien zawierać optymalny klaster czy też ile klastrów powinno być razem.

Ewaluacja spadku ryzyka HCOC

Miara spadku ryzyka względem klasyfikatora Cl^0 w korzeniu rozpatrywanego poddrzewa HCOC jest mierzona przy wykorzystaniu

$$\begin{aligned} E[X] &= \sum_{i=1}^{K}\sum_{j=1}^{K} E[X|true = C_i, predicted = C_j] \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K} P(C_i) m_{ij}^0 \left[\sum_{l=1}^{L} \frac{\sum_{t=1}^{K} m_{it}^0 f_{tl} \widetilde{m}_{ii}^l}{\sum_{l'=1}^{L}\sum_{j'=1}^{K} m_{ij'}^0 f_{j'l'}} - m_{ii}^0 \right], \end{aligned} \tag{2.35}$$

gdzie zmienna losowa $X = HCOC_{tr(x)}(x) - Cl_{tr(x)}(x)$ przedstawia wzrost wartości aktywacji na pozycji prawidłowej klasy $tr(x)$ dla wektora atrybutów x, K jest liczbą klas w danym poddrzewie, L liczbą klastrów, a $wide\tilde{m}_{ij}$ elementami ewaluacji macierzy pochodnych klasyfikatorów.

Wzrost aktywacji dla prawdziwej klasy (patrz rozdział 1.3) jest ściśle związany ze spadkiem ryzyka klasyfikatora HCOC. Maksymalizacja wyrażenia (2.35) odpowiada próbie maksymalizacji aktywacji na poprawnej pozycji wektora odpowiedzi $Cl_{tr(x)}(x)$.

2.6.6. Alternatywne klastrowanie dla lasu drzew decyzyjnych

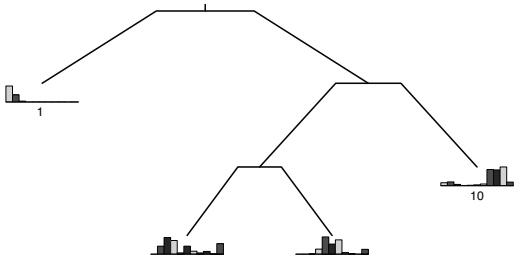
W trakcie budowy drzew decyzyjnych jak klasyfikatora w węźle HCOC każdy liść danego drzewa odpowiada jakiemuś klastrowi zawierającemu klasy w nim zawarte. Na rysunku 2.24 pokazane jest drzewo zbudowane dla problemu vowel. Utworzone liście odpowiadają następującym (nieobrobionym jeszcze) klastrom

$$\begin{aligned} Q^1 &= \{C_1, C_2, C_3\} \\ Q^2 &= \{C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{11}\} \\ Q^3 &= \{C_4, C_5, C_6, C_7, C_8, C_9, C_{11}\} \\ Q^4 &= \{C_1, C_2, C_3, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}\}. \end{aligned} \tag{2.36}$$

Dla każdego z nich potrzebne jest przyjęcie pewnego progu liczby przykładów danej klasy w liściu, powyżej której klasy będą zaliczane do węzła tak, aby utworzona macierz klastrowania była poprawna. To procedura analogiczna do tej przy klastrowaniu typu GNG. Może nią być wzięcie wszystkich klas powyżej mediany frakcji występowania w danym klastrze. Dzięki temu otrzymujemy następujące klastry

$$\begin{aligned} Q^1 &= \{C_1, C_2\} \\ Q^2 &= \{C_2, C_3, C_4, C_6, C_{11}\} \\ Q^3 &= \{C_5, C_6, C_7\} \\ Q^4 &= \{C_8, C_9, C_{10}\}. \end{aligned} \tag{2.37}$$

Po połączeniu tak znalezionych klastrów dla wszystkich drzew, procedura klastrująca wybiera te najczęściej występujące i łączy je w końcową macierz klastrowania F z zachowaniem warunków jej poprawności. Zaletą takiego podejścia jest natychmiastowe uzyskanie macierzy już w trakcie nauczania. Można jednak wykorzystać inne, przedstawione już, podejścia, które minimalizując pewne funkcje kosztu, powinny zwracać macierze dające lepszy klasyfikator HCOC.



Rysunek 2.24. Pojedyncze drzewo decyzyjne dla problemu vowel

2.6.7. Problem zapewnienia różnorodności klasyfikatorów

Składanie wyników z wielu klasyfikatorów będzie miało sens jedynie wtedy, gdy klasyfikatory będą działały różnie, a w szczególności popełniały różne błędy (Kumar, Ghosh, 1999; Shipp, Kuncheva, 2002; Kuncheva, 2005; Giacinto, Roli, 2001; Meynet, Thiran, 2010). Jednak sama różnorodność nie jest wystarczająca i zbiór klasyfikatorów zależnych może być lepszy od zbioru klasyfikatorów niezależnych, ale i słabszy od najgorszego z zestawu (Shipp, Kuncheva, 2002). W dodatku A.10 opisane są różne miary różnorodności.

Różnorodność trzeba także zapewnić w HCOC. Konieczne jest zagwarantowanie różnorodności wśród klasyfikatorów pochodnych, w szczególności dla obszarów, gdzie klastry się nakładają. To zadanie będzie wymagało pewnej interakcji przy nauczaniu klasyfikatorów będących potomkami tego samego węzła.

Najlepiej zapewnić różnorodność już na poziomie klastrowania. Jest to możliwe przez użycie odpowiedniej funkcji dopasowania wykorzystywanej w trakcie tworzenia macierzy F (Podolak, Roman, 2011b). Nie jest to zadanie proste, ponieważ bezpośrednia maksymalizacja miary różnorodności niekoniecznie musi prowadzić do optymalnych grup klasyfikatorów, a sama redukcja korelacji miedzy klasyfikatorami nie jest wystarczająca (Kumar *et al.*, 2002; Meynet, Thiran, 2010).

W modelu HCOC dla uzyskania różnorodności w danym problemie możliwe są oczywiście metody polegające na utworzeniu nadmiarowej liczby klasyfikatorów dla każdego z węzłów pochodnych, a następnie wyborze spośród nich grupy tych, które maksymalizują wybraną miarę. Alternatywnie, w trakcie równoległe-

go nauczania klasyfikatorów w potomkach danego węzła, można jednocześnie z nauczaniem ewaluować miarę różnorodności, aby przerywać nauczanie, gdy funkcja celu osiągnie wysoką wartość. Jednak obie te metody potrzebują wielu zasobów, są więc nieopłacalne.

Możliwe jest jednak zaprojektowanie funkcji dopasowania w modelach klastrowania. Algorytm klastrowania Bayesowskiego (patrz rozdział 2.6.2) buduje macierz klastrowania F poprzez sklejanie mniejszych klastrów, tak aby maksymalizować zadaną funkcję. Może to być ślad macierzy generalizacji G (patrz definicja 2.24), zwykła miara poprawności macierzy F, miara wykorzystująca ewaluację macierzy błędnych klasyfikacji dla nienauczonych jeszcze klasyfikatorów potomnych (patrz rozdział 1.2.2), wykorzystanie ewaluacji ryzyka dla HCOC (patrz rozdział 2.7) lub jakiejś kombinacji tych miar. Można też uwzględnić w tych miarach współczynnik różnorodności.

Kontrola różnorodności klasyfikatorów pochodnych

Opisane modele kontroli różnorodności zakładają zwykle możliwość sprawdzenia klasyfikacji dla każdego przykładu przez każdy model potomny. Wykorzystując jednak ewaluację macierzy, można przybliżyć *łączną* wartość różnorodności dla danego klasyfikatora zdefiniowanego przez macierz pomyłek M klasyfikatora w korzeniu i aktualnej macierzy klastrowania F

$$\begin{aligned} diversity(M,F) &= \frac{1}{\binom{L}{2}} \sum_{l_1<l_2} d(\widetilde{M}^{l_1}, \widetilde{M}^{l_2}) \\ &= \frac{1}{\binom{L}{2}} \sum_{l_1<l_2} \sum_i \sum_j \frac{|\widetilde{m}_{ij}^{l1} \widetilde{m}_{ij}^{l2}|}{\widetilde{m}_{ij}^{l1} + \widetilde{m}_{ij}^{l2}}, \end{aligned} \tag{2.38}$$

które jest sumą różnic macierzy pomyłek dla każdej pary klasyfikatorów l_1 i l_2 i dla każdej pary indeksów klas i oraz j.

Doświadczenie 2.30. *Model HCOC wykorzystuje dla zadania klastrowania wiele różnych funkcji dopasowania wykorzystywanych przez wszystkie podejścia do klastrowania. Zdefiniowane funkcje wykorzystują kombinacje liniowe różnych miar: ewaluacji funkcji ryzyka typu* pseudo-loss *(patrz definicja 1.13) z wykorzystaniem aproksymacji macierzy pomyłek $\widetilde{M}$ klasyfikatorów pochodnych (patrz wzór (1.18)), ewaluacji ryzyka z twierdzenia 2.31, ewaluacji $\widetilde{M}$ z wzoru (1.17), miary poprawności macierzy klastrowania F z definicji 2.11 oraz miary różnorodności dwupoziomowego HCOC (2.38) z ewaluowanymi $\widetilde{M}$. Tabela 2.4 pokazuje wyniki eksperymentów. Dla każdego spośród wykorzystanych* 11 *zbiorów uczących, miary* fitness *zostały uporządkowane według zwracanych średnich wyników. Średnia ranga w tabeli odpowiada średniej pozycji na tej liście, co pozwala porównać różne algorytmy z wykorzystaniem różnych danych (Demŝar, 2006).*

Tabela 2.4 pokazuje, że większość funkcji dopasowania daje porównywalne wyniki. Odstają od nich `fitness.C` (pseudo-loss, *diversity i poprawność macierzy F),* `fitness.G`

Tabela 2.4. Porównanie różnych funkcji *fitness*. Wyniki uzyskane przez wielokrotne nauczanie na zbiorach typu *bootstrap* i wyliczenie błędu typu $Err^{(0.632)}$ (patrz A.13). Miary `fitness.E` i `fitness.H` różnią się stosunkiem uwzględnionych parametrów: w `fitness.E` jest większy udział miary różnorodności. Dla każdej miary wyniki policzone były metodą *bootstrap* dla 8 różnych zbiorów uczących, dwóch algorytmów klastrujących i różnych parametrów definiujących złożoność klasyfikatorów w węzłach

fitness	pseudo	F	diversity	$\widehat{R}[HCOC]$	$trace(G)$	średnia ranga
A	✓	✓				3.91
B		✓	✓			3.02
C	✓	✓	✓			5.44
D		✓		✓		3.22
E		✓	✓	✓		3.31
G		✓	✓		✓	4.27
H		✓	✓	✓		3.08

(ślad macierzy generalizacji G i poprawność F) oraz `fitness.A` (pseudo-loss *i poprawność F) – tu widać nieadekwatność podejścia do HCOC przez koszt* pseudo-loss. *Najlepiej sprawują się te miary dopasowania, które biorą pod uwagę oczekiwany koszt klasyfikatora HCOC pochodzący z twierdzenia 2.31 lub miarę różnorodności. Kombinacja* pseudo-loss *i różnorodności wydaje się prowadzić do sprzeczności, a stąd do słabych wyników.*

2.7. Zbieżność nauczania HCOC

Wagi $w_l(x)$ modelu HCOC obliczane są proporcjonalnie do odpowiedzi klasyfikatora w korzeniu i w zależności od tego, które klasy należą do danego klastra l. Podstawowym zagadnieniem jest określenie, czy w zbudowanym tak modelu, przy dodawaniu kolejnych warstw klasyfikatorów, można liczyć na zmniejszanie się błędu. Twierdzenia w następnych częściach pokazują, jakie warunki powinny być spełnione, aby tak się stało.

2.7.1. HCOC jako złożony klasyfikator

HCOC składa się z korzenia oraz szeregu klasyfikatorów potomnych, będących korzeniami kolejnych HCOC rozwiązujących podproblemy, które powinny być prostsze do rozwiązania (Frank *et al.*, 2003; Hastie *et al.*, 2001). Problem, w jaki

sposób należy składać wyniki, od dawna jest jednym z centralnych zagadnień projektowania złożonych systemów klasyfikacji, a wnioski w dużym stopniu zależą od architektury modelu i postaci klasyfikatorów w węzłach (patrz m.in. Kulkarni, 1978; Schuerman, Doster, 1984; Kuncheva, 2000; Kumar *et al.*, 2002; Duch, Itert, 2003; Cesa-Bianchi *et al.*, 2006; Ciampi *et al.*, 2007; Meynet, Thiran, 2010).

Rozpatrzmy na początku najprostszą architekturę HCOC, którą jest dwupoziomowy model z korzeniem i jedną warstwą klasyfikatorów pochodnych.

Twierdzenie 2.31. *(Podolak, Roman, 2012) Niech HCOC będzie dwupoziomowym klasyfikatorem. Niech korzeń Cl^0 będzie słaby. Niech $\ell()$ będzie kwadratową funkcją kosztu. Niech HCOC wykorzystuje standardowe typy wag i ewaluuje wszystkie poddrzewa algorytmem* All-subtrees.

Wartość funkcji ryzyka dla HCOC jest niższa niż dla Cl^0, pod warunkiem że

$$2\sum_{i=1}^{K} p_i m_{ii}^0 - \sum_{i=1}^{K} p_i \sum_{k=1}^{K} (m_{ik}^0)^2 - \frac{2}{\max\limits_{k'} \sum_{l'} f_{k'l'}^0} \sum_{k=1}^{K} \sum_{l=1}^{L^0} \sum_{i=1}^{K} p_i f_{kl}^0 m_{ii}^l m_{ik}^0 + 1 \tag{2.39}$$

jest ujemne.

Dowód. Będziemy aproksymować wartości oczekiwane, wykorzystując macierze pomyłek M^0 w korzeniu i $M^1, \ldots, M^L$ w klasyfikatorach następnej warstwy. Niech $\widehat{y}(x)$ będzie K-elementowym binarnym wektorem z wartością 1 na pozycji poprawnej klasy wektora x. Oznaczamy indeks tej klasy jako $tr(x)$. Niech $\mathcal{X}_i = \{x \in \mathcal{X} : (x, C_i) \in \mathcal{D}\}$. Dla kwadratowej funkcji kosztu $\ell(x, \widehat{y}(x), y(x)) = \sum_{k=1}^{K} (y_k^0 - \widehat{y}_k(x))^2$ mamy następującą wartość funkcji ryzyka klasyfikatora Cl^0 w korzeniu:

$$\begin{aligned}
R(Cl^0) = E[\ell(Cl^0)] &= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} \ell(x, \widehat{y}(x), y^0(x)) \\
&= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} \left[\sum_{k=1}^{K} y_k^0(x)^2 - 2 \sum_{k=1}^{K} y_k^0(x) \widehat{y}_k(x) + \sum_{k=1}^{K} \widehat{y}_k(x)^2 \right] \\
&= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} \sum_{k=1}^{K} y_k^0(x)^2 - 2 \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} y_i^0(x) + \sum_{i=1}^{K} p_i \\
&= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} \sum_{k=1}^{K} y_k^0(x)^2 - 2 \sum_{i=1}^{K} p_i m_{ii}^0 + 1, \qquad (2.40)
\end{aligned}$$

gdzie m_{ii}^0 jest wartością oczekiwaną poprawnej klasyfikacji przez Cl^0 przykładów z klasy C_i.

Dla HCOC mamy analogicznie

$$\begin{aligned}
E[\ell(HCOC)] &= \sum_{x \in X} \ell(x, \widehat{y}(x), y^{HCOC}(x)) \\
&= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} \sum_{k=1}^{K} y_k^{HCOC}(x)^2 - 2 \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} y_i^{HCOC}(x) + 1. \qquad (2.41)
\end{aligned}$$

Zauważmy, że $\sum_{k=1}^{K} y_k^{HCOC}(x) = 1$, stąd $\sum_{k=1}^{K} y_k^{HCOC}(x)^2 \leq 1$.

$$\begin{aligned} E[\ell(HCOC)] &= \sum_{i=1}^{K} p_i \frac{1}{X_i} \sum_{x \in X_i} \left[\sum_{k=1}^{K} y_k^{HCOC}(x)^2 - 2y_i^{HCOC}(x) + 1 \right] \\ &\leq \sum_{i=1}^{K} p_i \frac{1}{X_i} \sum_{x \in X_i} \left[1 - 2y_i^{HCOC}(x) + 1 \right] \\ &= 2 \sum_{i=1}^{K} p_i \frac{1}{X_i} \sum_{x \in X_i} 1 - \sum_{i=1}^{K} p_i \frac{1}{X_i} \sum_{x \in X_i} y_i^{HCOC}(x) \\ &= 2 - 2 \sum_{i=1}^{K} p_i \frac{1}{X_i} \sum_{x \in X_i} y_i^{HCOC}(x). \end{aligned} \tag{2.42}$$

Równość zachodzi tylko w przypadku, gdy HCOC, dla wszystkich x, zwraca wektor aktywacji $y^{HCOC}(x)$ z dokładnie jedną jedynką (niekoniecznie w pozycji prawdziwej klasy) oraz $K-1$ zerami.

Dla dwupoziomowego HCOC końcowy wektor prawdopodobieństwa klas, przy ewaluacji metodą All-subtrees, ma postać

$$y_i^{HCOC}(x) = \sum_{l: V^l \in child(V^0); C_i \in Q^l} w_l^0(x) y_i^l(x), \tag{2.43}$$

gdzie $w_l^0(x)$ jest wagą l-tego klasyfikatora pochodnego. Waga jest obliczana *niezależnie* dla każdego wektora atrybutów x (metoda zależna od danych)

$$w_l^0(x) = \sum_{k=1}^{K} f_{kl}^0 y_k^0(x) \Big/ \sum_{l': V^{l'} \in child(V^0)} \sum_{k'=1}^{K} f_{k'l'}^0 y_{k'}^0(x). \tag{2.44}$$

Wektor $y^l(x)$ to aktywacja l-tego pochodnego klasyfikatora, $y^0(x)$ to aktywacja klasyfikatora w korzeniu. Wstawiając (2.44) do (2.43), otrzymujemy

$$y_i^{HCOC}(x) = \sum_{l: V^l \in child(V^0)} \frac{\sum_{k=1}^{K} y_i^l(x) f_{kl}^0 y_k^0(x)}{\sum_{l'=1}^{L^0} \sum_{k'=1}^{K} f_{k'l'}^0 y_{k'}^0(x)}. \tag{2.45}$$

Wystarczy teraz pokazać, kiedy $E[\ell(HCOC)] < E[\ell(Cl^0)]$. Uwzględniamy pewne cechy poszczególnych klasyfikatorów (w korzeniu i w następnej warstwie) oraz własności macierzy klastrowania F. Po pierwsze, ograniczenie górne oczekiwanego kosztu HCOC

$$\begin{aligned} E[\ell(HCOC)] &= 2 - 2 \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x \in X_i} y_i^{HCOC}(x) \\ &= 2 - 2 \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{l=1}^{L^0} \frac{\sum_{k=1}^{K} y_i^l(x) f_{kl}^0 y_k^0(x)}{\sum_{k'=1}^{K} \sum_{l'=1}^{L^0} f_{k'l'}^0 y_{k'}^0(x)}. \end{aligned} \tag{2.46}$$

Zauważmy, że mianownik

$$\sum_{k'=1} y^0_{k'}(x)^K \sum_{l'=1}^{L^0} f^0_{k'l'} \leq B \sum_{k'=1}^{K} y^0_{k'}(x) \tag{2.47}$$

dla pewnego $B = \max_{k'} \sum_{l'=1}^{L^0} f^0_{k'l'}$ (oczywiście $\sum_{k'=1}^{K} y^0_{k'}(x) = 1$), które odpowiada największemu nakładaniu się klas, czyli liczbie klastrów, do których należy pojedyncza klasa. Stąd

$$\begin{aligned}
E[\ell(HCOC)] &= 2 - 2\sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x\in X_i} \sum_{l=1}^{L^0} \frac{\sum_{k=1}^{K} y^l_i(x) f^0_{kl} y^0_k(x)}{\sum_{k'=1}^{K}\sum_{l'=1}^{L^0} f^0_{k'l'} y^0_{k'}(x)} \\
&\leq 2 - \frac{2}{B}\sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x\in X_i} \sum_{l=1}^{L^0}\sum_{k=1}^{K} y^l_i(x) f^0_{kl} y^0_k(x) \\
&= 2 - \frac{2}{B}\sum_{i=1}^{K} p_i E\left[\sum_{l=1}^{L^0}\sum_{k=1}^{K} y^l_i(x) f^0_{kl} y^0_k(x) | x \in X_i\right] \\
&= 2 - \frac{2}{B}\sum_{i=1}^{K} p_i \sum_{l=1}^{L^0}\sum_{k=1}^{K} f^0_{kl} E[y^l_i(x)|x\in X_i]\; E[y^0_k(x)|x\in X_i] \qquad (2.48) \\
&= 2 - \frac{2}{B}\sum_{i=1}^{K}\sum_{l=1}^{L^0}\sum_{k=1}^{K} f^0_{kl} m^l_{ii} m^0_{ik}, \qquad (2.49)
\end{aligned}$$

ponieważ dla danej macierzy błędnych klasyfikacji klasyfikatora Cl^0 w korzeniu $M^0 = (m^0_{ij})^K_{i,j=1}$, możemy przybliżyć wartość oczekiwaną $E[y^0_k(x)|x \in X_i] = m^0_{ik}$. Równość w (2.48) wynika z obserwacji, że klasyfikatory Cl^0 i Cl^l są niezależne.

Dolne ograniczenie dla $E[\ell(Cl^0)]$

$$\begin{aligned}
E[\ell(Cl^0)] &= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x\in X_i}\sum_{k=1}^{K} y^0_k(x)^2 - 2\sum_{i=1}^{K} p_i \frac{1}{|X_i|}\sum_{x\in X_i} y^0_i(x) + 1 \\
&= \sum_{i=1}^{K} p_i \frac{1}{|X_i|} \sum_{x\in X_i}\sum_{k=1}^{K} y^0_k(x)^2 - 2\sum_{i=1}^{K} p_i E[y^0_i(x)|x\in X_i] + 1 \\
&\geq \sum_{i=1}^{K} p_i \sum_{k=1}^{K} E[y^0_k(x)]^2 - 2\sum_{i=1}^{K} p_i m^0_{ii} + 1 \\
&= \sum_{i=1}^{K} p_i \sum_{k=1}^{K} (m^0_{ik})^2 - 2\sum_{i=1}^{K} p_i m^0_{ii} + 1, \qquad (2.50)
\end{aligned}$$

ponieważ $E[y^0_k(x)^2] \geq E[y^0_k(x)]^2$.

Tak więc, aby HCOC miał niższy błąd, następujące wyrażenie musi być ujemne

$$E[\ell(HCOC)] - E[\ell(Cl^0)]$$

$$= \underbrace{2\sum_{i=1}^{K} p_i m_{ii}^0}_{\Delta_1} - \underbrace{\sum_{i=1}^{K} p_i \sum_{k=1}^{K} (m_{ik}^0)^2}_{\Delta_2} - \underbrace{\frac{2 \cdot \sum_{k=1}^{K} \sum_{l=1}^{L^0} \sum_{i=1}^{K} p_i f_{kl}^0 m_{ii}^l m_{ik}^0}{\max_{k'} \sum_{l'=1}^{L^0} f_{k'l'}^0}}_{\Delta_3} + 1. \tag{2.51}$$

□

Warunek (2.51) jest wyrażony poprzez wartości oczekiwane funkcji kosztu. Każda z jego składowych wskazuje na pewien aspekt nauczania i tworzenia macierzy klastrowania:

- Δ_1 jest proporcjonalna do dokładności klasyfikatora Cl^0 w korzeniu i wskazuje na to, że łatwiej osiągnąć spadek błędu, jeśli klasyfikator w korzeniu jest słaby;
- Δ_2 jest sumą kwadratów wszystkich elementów macierzy pomyłek dla Cl^0: składnik Δ_2 osiąga największe wartości, gdy dla danego i wartości m_{ik}^0 są skupione wokół kilku indeksów k, czyli gdy klasa C_i jest mylona jedynie z kilkoma innymi klasami; jest to szczególnie ważne dla klas C_i o wysokim prawdopodobieństwie *a priori* $p_i = P(C_i)$;
- Δ_3 opisuje interakcję między klasyfikatorem nadrzędnym Cl^0 i pochodnymi Cl^l
  - wartość mianownika $\max_{k'} \sum_{l'=1}^{L} f_{k'l'}^0$ to maksymalna liczbie klastrów, do których należy pojedyncza klasa; aby wyrażenie (2.51) osiągało minimalną wartość, klasy powinny być równomiernie rozłożone między klastrami, a wobec tego pojedyncze klasy nie powinny należeć do większości (w szczególności wszystkich) klastrów;
  - licznik Δ_3 odpowiada sytuacji, gdy przykłady z prawdziwej klasy C_i są błędnie klasyfikowane jako C_k; wysoka wartość może być uzyskana dzięki wysokim wartościom m_{ii}^l, które odpowiadają prawdopodobieństwu dobrej klasyfikacji w klasyfikatorach pochodnych; w chwili klastrowania nie mamy jednak bezpośredniego wpływu na m_{ii}^l. Można to ewentualnie osiągnąć przez założenie podobieństwa klasyfikatorów pochodnych do klasyfikatora w korzeniu, stąd estymować ich błąd, wbudowując tę informację do funkcji kosztu klastrowania (Podolak, Roman, 2011c), inną drogą jest zbudowanie takiej macierzy klastrowania, aby C_k należało do wielu klastrów, w których jest także C_i. Jest to oczywistym założeniem klastrowania (patrz rozdział 2.2.1).

Postać warunku (2.51) dla funkcji kosztu typu $\ell(x, y, Cl(x)) = 1 - Cl_{tr(x)}(x)$, zamiast kwadratowej, jest analogiczna i daje analogiczne wnioski.

Twierdzenie 2.32. *Dla funkcji kosztu* $\ell(x, y, Cl^{HCOC}(x)) = 1 - Cl^{HCOC}_{tr(x)}(x)$ *funkcja ryzyka* $R[HCOC]$ *jest ograniczona*

$$R[HCOC] \leq 1 - \frac{1}{\max\limits_{k'} \sum_{l'=1}^{L} f_{k'l'}} \sum_{i=1}^{K} p_i \sum_{l=1}^{L} \sum_{k=1}^{K} f_{kl} m^l_{ii} m^0_{ik}. \tag{2.52}$$

2.7.2. Błąd HCOC a słabość klasyfikatorów bazowych

Możliwe jest także wykorzystanie założenia, że klasyfikator w korzeniu jest słaby. Wymaga to jednak „silniejszego" założenia o słabości w sensie definicji 1.21, a więc że Cl^0 jest słaby dla każdej klasy niezależnie

$$E[Cl^0_i(x)|tr(x) = i] = \alpha(K) + \epsilon_i \tag{2.53}$$

dla każdego $i = 1, \ldots, K$, gdzie $\epsilon > 0$.

Twierdzenie 2.33. *Niech* Cl^0 *spełnia (2.53) dla każdej klasy. Niech HCOC składa się z korzenia* Cl^0 *i potomków* $Cl^1, \ldots, Cl^L$.

Oczekiwana wartość odpowiedzi dla prawidłowych klas klasyfikatora HCOC jest wyższa niż Cl^0, *tzn.* $E[HCOC] > E[Cl^0]$, *jeśli*

$$\sum_{l=1}^{L} \left(\alpha(K) + (|Q_l| - 1) \cdot \frac{1 - \alpha(K)}{K - 1} \right) |Q_l| \cdot \alpha(|Q_l|) \pi K(\alpha(K) + \epsilon), \tag{2.54}$$

gdzie $\pi = \max\limits_{i} \{\sum_{l=1}^{K} f_{il} E[w_l|tr(x) = i]\}$ *oraz* $\epsilon = \sum_i \epsilon_i$.

Z tego twierdzenia wynikają następujące zależności, które będą wspomagać ograniczenie błędu HCOC:

- model daje mniejszy błąd dla wielu klastrów (ale, z drugiej strony, może to być bardziej kosztowne obliczeniowo),
- klastry powinny zawierać dużą liczbę klas (ten wniosek, w połączeniu z poprzednim, sugeruje duże nakładanie się klastrów),
- wagi w_l powinny być równomiernie rozłożone.

2.7.3. Zależność błędu HCOC od błędu generalizacji

Błąd HCOC zależy w dużym stopniu od sposobu i skuteczności klastrowania, a więc od błędu klasyfikatora uogólnionego wskazującego poprawną klasę. Jego funkcja kosztu (patrz definicja 2.24) zdefiniowana została jako

$$\ell(x, trQ(x; F), V(x)) = 1 - V(x) \cdot trQ(x; F(x))^T,$$

gdzie $trQ(x;F) = [f_{tr(x),1}, \ldots, f_{tr(x),L}]$ oznacza wzorzec poprawnego przypisania przykładu x do klastrów przy wykorzystaniu macierzy klastrowania F. $V(x)$ odpowiada klasyfikacji x przez węzeł V, tzn. przez zawarty w nim klasyfikator bazowy Cl i macierz F.

Uwzględniając to, można zdefiniować empiryczne ryzyko *HCOC* (dla uproszczenia zapisu, ograniczamy się do HCOC składającego się tutaj jedynie z korzenia i jednego poziomu poniżej) jako

$$R[HCOC] = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L}\ell(x, trQ(x;F), V(x))\ell(x, C_{tr(x)}, Cl^l(x)), \tag{2.55}$$

gdzie $Cl^l(x)$ są odpowiedziami klasyfikatorów w węzłach pierwszego poziomu (jeśli zastąpić Cl^l przez $HCOC^l$ oznaczające odpowiedź kolejnego modelu HCOC dla podproblemu, to otrzymamy ogólny opis dla dowolnej liczby poziomów).

Twierdzenie 2.34. (uogólnienie Podolak, Bartocha (2009)) *Niech HCOC będzie dwupoziomowym modelem z klasyfikatorem Cl^0. Niech $R[Cl^0] < \beta - \epsilon$ dla $0 < \epsilon$. Niech klasyfikator uogólniony ma błąd generalizacji mniejszy niż δ i niech klasyfikatory pochodne mają oczekiwaną wartość kosztu*

$$R[Cl^i] < 1 - \beta - \epsilon - \mu, \tag{2.56}$$

dla $\mu > 0$. HCOC ma błąd mniejszy od Cl^0, pod warunkiem że

$$\mu > \frac{\delta(1 - \beta + \epsilon)}{1 - \delta}. \tag{2.57}$$

Dowód. Ryzyko dla dwupoziomowego HCOC można rozbić na dwa składniki: pochodzące od dobrego przypisania do klastrów i do błędnego

$$\begin{aligned} R[HCOC] = &\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} trQ(x;F)P(C_k)P(x)f_{tr(x),l} \\ &+ \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} trQ(x;F)P(C_k)P(x)(1 - f_{tr(x),l}). \end{aligned} \tag{2.58}$$

Załóżmy, dla prostoty, równomierny rozkład danych, stąd współczynnik $1/N$ (poprawnie będzie używać $P(x)$). Wartość $f_{tr(x),l}$ jest z definicji równa 1, jeśli prawdziwa klasa przykładu x należy do klastra Q^l. Takie rozbicie jest uprawnione, gdyż klasyfikacja od każdego klastra będzie sumowana i tak tylko raz. Pierwszy składnik odpowiada klasyfikacjom od poprawnych klastrów, jest więc związany z klasyfikacją przez klasyfikator uogólniony, którego jedynym zadaniem jest wskazywanie poprawnych klastrów.

Załóżmy, że klasyfikator uogólniony ma ryzyko mniejsze od pewnego δ większego od zera. Wtedy

$$\begin{aligned} R[HCOC] =&(1-\delta)\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} trQ(x;F)P(C_k)P(x)f_{tr(x),l} \\ &+\delta\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} trQ(x;F)P(C_k)P(x)(1-f_{tr(x),l}). \end{aligned} \tag{2.59}$$

Poprawna klasa danego x nie należy do klastrów wybieranych w drugim składniku 2.58, stąd klasyfikacja jest tam zawsze niepoprawna i sumuje się do jedności, wobec czego

$$R[HCOC] = (1-\delta)\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} trQ(x;F)P(C_k)P(x)f_{tr(x),l} + \delta. \tag{2.60}$$

Z założenia twierdzenia klasyfikator w korzeniu jest słaby i ma błąd $R[Cl] < \beta - \epsilon$. Klasyfikatory w węzłach pochodnych także są słabe. Pytamy się, o jaki czynnik μ muszą mieć błąd $R[Cl^l] < \beta - \epsilon - \mu$ mniejszy, aby błąd HCOC był mniejszy od błędu klasyfikatora w korzeniu.

$$R[HCOC] = (1-\delta)(\beta - \epsilon - \mu) + \delta, \tag{2.61}$$

$$R[HCOC] < R[Cl] < \beta - \epsilon. \tag{2.62}$$

Aby ten warunek był spełniony, podstawiamy (2.61) do (2.62), uzyskując

$$\mu > \frac{\delta(1-\beta+\epsilon)}{(1-\delta)}. \tag{2.63}$$

□

Im mniejsze będą podproblemy, tym prościej ten warunek będzie możliwy do osiągnięcia. Stąd potrzeba podziału problemu na mniejsze podproblemy. Wartość μ pokazuje, na ile silnie powinny być nauczone klasyfikatory w kolejnej warstwie, aby błąd całego modelu malał.

2.8. Podsumowanie i uwagi

W tym, centralnym dla pracy, rozdziale zdefiniowany został formalnie model klasyfikatora HCOC. Pokazano jego główne metody takie jak zasady tworzenia bazowych klasyfikatorów w węzłach oraz metody generowania podziału na podproblemy. Zdefiniowane zostały różne algorytmy implementujące te zadania, wraz z dyskusją na temat ich poprawności i skuteczności. Dla algorytmów klastrowania zdefiniowano miary dopasowania pozwalające na skuteczne tworzenie poprawnych klastrów, jednocześnie zwiększając ich różnorodność.

Zdefiniowano metody ewaluacji wyników klasyfikacji. Podane zostały alternatywne metody obliczania wag klasyfikatorów pochodnych oraz alternatywne reguły agregacji wyników, wraz z uzasadnieniem. Pokazano, w jaki sposób HCOC korzysta z tych definicji dla uzyskania lepszej skuteczności. Tu szczególnie ciekawe są metody ewaluacji Restricted i α-Restricted, które ograniczają złożoność obliczeniową, jednocześnie zmniejszając błąd względem metody All ewaluującej wszystkie ścieżki czy metody Single-Path podążającej jedynie ścieżką najbardziej prawdopodobną.

Pokazano, iż zdefiniowane metody obliczania wag skutkują tym, że wagi klastrów, które zawierają poprawną dla przykładu x klasę, będą statystycznie wyższe od wag tych klastrów, które jej nie zawierają. W oczywisty sposób zwiększa to prawdopodobieństwo uzyskania prawidłowej odpowiedzi.

Na koniec pokazano, że przy zachowaniu pewnych warunków błąd klasyfikatora HCOC będzie malał wraz z dodawaniem kolejnych poziomów.

Rozdział 3

Eksperymenty i doświadczenia

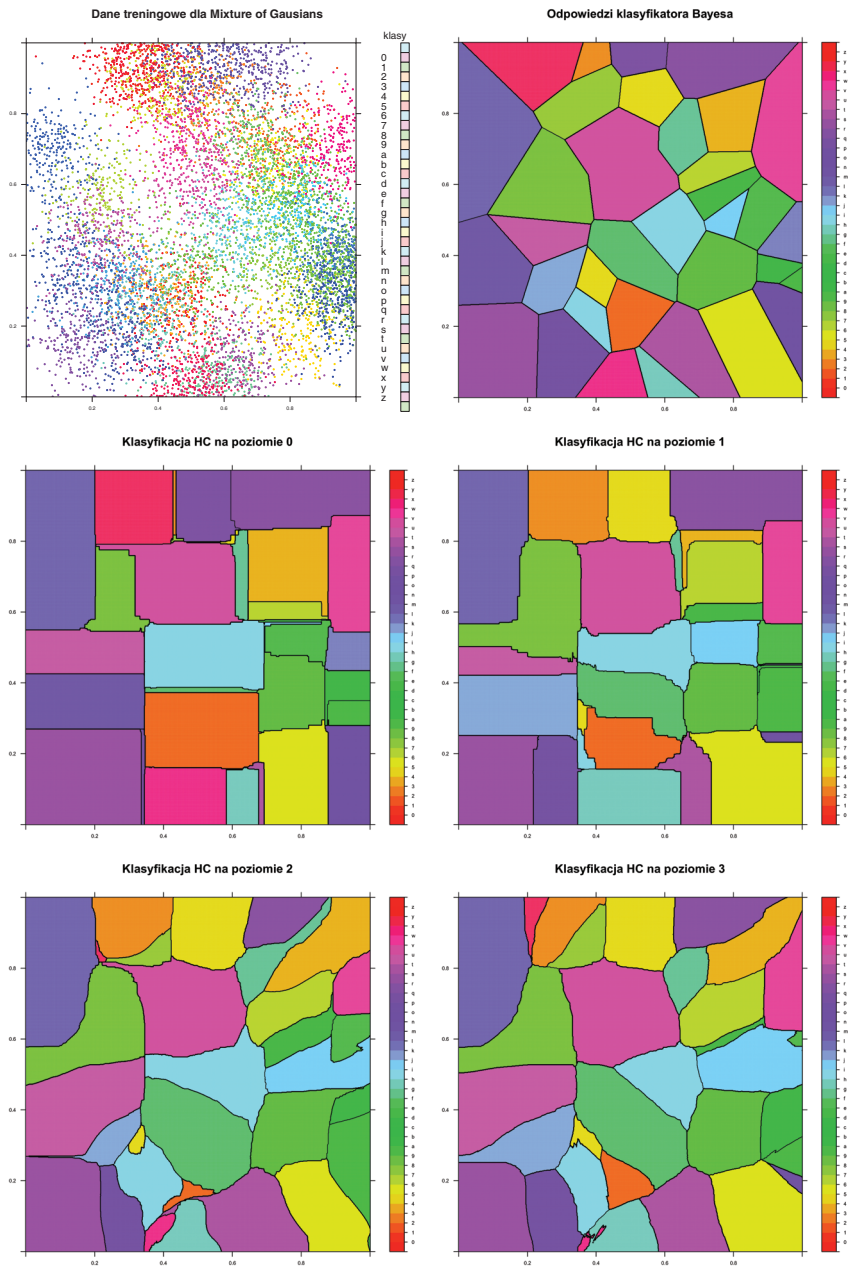
W trakcie pracy nad modelem HCOC wykonane zostały różne implementacje. Najpierw była to implementacja w języku Java, pozwalająca na urównoleglenie szeregu procesów. W tej implementacji dla tworzenia klasyfikatorów bazowych zastosowany został model sieci neuronowych oraz algorytm nauczania Resilient Propagation. Wykorzystywane było środowisko WEKA ® (Witten, Frank, 2000).

W dalszym ciągu, ze względu na szybkość tworzenia samej implementacji, stworzona została nowa przy wykorzystaniu pakietu statystycznego R ® i wielu związanych z nimi modułów (R Development Core Team, 2005). Ta implementacja jest obiektowa. Jako klasyfikatory bazowe wykorzystane są lasy drzew decyzyjnych oraz sieci neuronowe nauczane algorytmem BFGS (Broyden–Fletcher–Goldfarb–ShannoSchraudolph *et al.* (2007)).

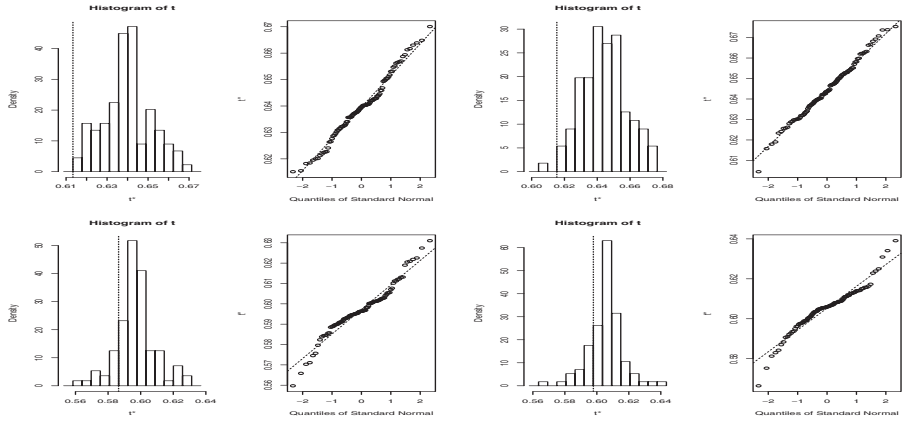
3.1. Eksperyment Mixture of Gaussians dla wielu klas wyjściowych

Podstawowym celem architektury HCOC jest rozwiązanie problemu klasyfikacji dla zadań z dużą liczbą klas. Podstawowym podejściem do tego typu zadań jest podział na szereg podproblemów, rozwiązanie każdego z osobna, w końcu scalenie wyników. W HCOC podział rozpoczyna się od budowy słabego klasyfikatora, tak aby poprzez analizę jego pracy wygenerować podział zadania na podproblemy.

Już wcześniej pokazane zostały wyniki doświadczenia dla problemu Mixture of Gaussians dla niewielkiej jednak liczby 10 klas wyjściowych. Na rysunku 3.1 pokazane jest rozwiązanie problemu dla 36 klas wyjściowych. Dane zostały wylosowane z niezależnych dwuwymiarowych rozkładów normalnych na kwadracie $[0,1]^2$ i wariancji $\sigma_{xx} = \sigma_{yy} = 0.00632 \pm \epsilon$. Pokazany naiwny klasyfikator Bayesowski daje rozwiązanie minimalizujące koszt przy tak założonych rozkładach *a priori*.



Rysunek 3.1. HCOC dla syntetycznego problemu złożonego z 36 gausowskich chmur. Wszędzie ewaluacja typu RESTRICTED oraz klastrowanie Bayesowskie



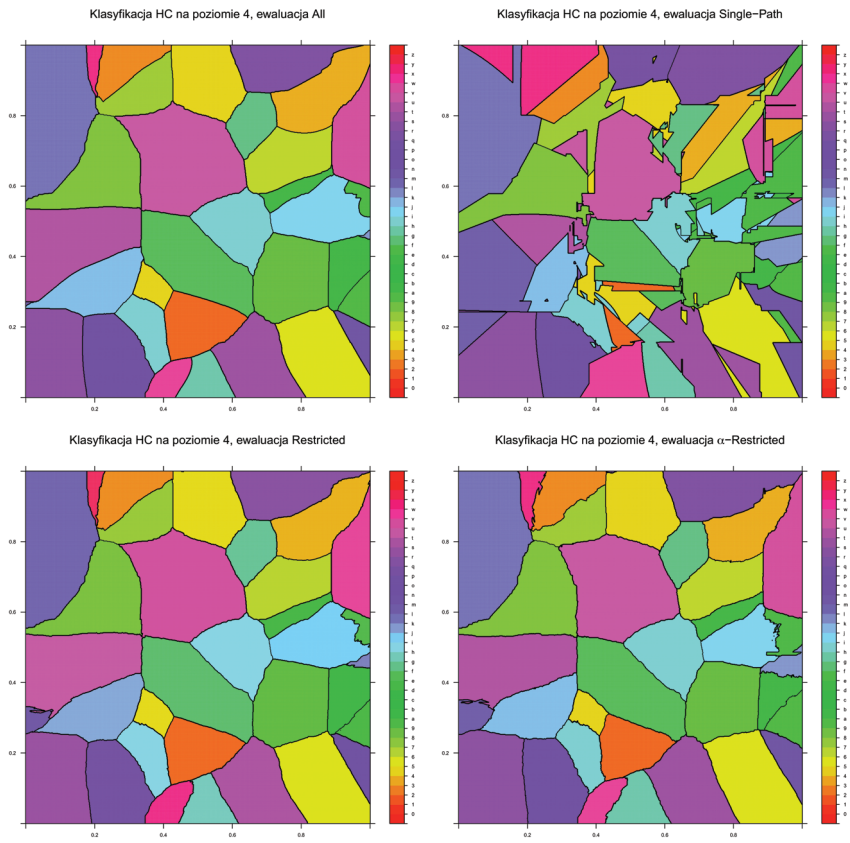
Rysunek 3.2. Histogramy i wykresy kwantylowe błędów dla rozwiązań problemu Mixture of Gaussians dla kolejno 0, 1, 2 i 3 poziomu. Wyniki uzyskane przez stukrotne powtórzenie eksperymentu typu *bootstrap*

Model HCOC wykorzystywał klasyfikatory bazowe dwóch typów: lasy drzew decyzyjnych, gdy podproblem miał więcej niż 11 klas wyjściowych, oraz sieci neuronowe w przeciwnym wypadku. Drzewo modelu składało się z korzenia i 4 poziomów poniżej. Na rysunku 3.1 nie są pokazane wyniki dla ostatniego poziomu, który zresztą nie był w pełni wypełniony (duża część podproblemów kończyła się na poziomie trzecim). Rysunki dają jednak dobre przybliżenie szczególnie w obszarach problematycznych klasyfikacji.

Ze względu na wykorzystanie lasu, a nie pojedynczego drzewa decyzyjnego dla podziału w korzeniu (poziom 0), nie wszystkie podziały są równoległe do osi. Na poziomie 1 (pierwszy poniżej korzenia) model w większości wykorzystuje lasy drzew, jednak kilka podproblemów wykorzystuje sieci neuronowe, stąd często nierównoległe do osi podziały. Na ostatnich poziomach są to praktycznie zawsze sieci neuronowe.

Trzeba zauważyć, że *wszystkie* wykorzystane w rozwiązaniu sieci neuronowe w węzłach miały tylko po dwa neurony ukryte i każda była uczona jedynie przez 50 iteracji algorytmu quasi-newtonowskiego BFGS (tylko klasyfikatory w liściach były douczane przez dodatkowe 100 iteracji). Bardzo zbliżone wyniki możliwe były do uzyskania przy mniejszej o połowę liczbie iteracji. Histogramy na rysunku 3.2 pokazują, jak rozkładają się błędy dla kolejnych poziomów klasyfikatora HCOC. Widać, że wraz ze wzrostem głębokości rozkład jest coraz bardziej zbliżony do rozkładu normalnego. Nie ma dużych odstępstw od rozkładu normalnego.

Rysunek 3.3 pokazuje wyniki różnych ewaluacji dla tego problemu. Widać, jak różnią się klasyfikacje:



Rysunek 3.3. Różne typy ewaluacji syntetycznego problemu 36 gausowskich chmur. Od góry do dołu, od lewej do prawej ewaluacje All, Single-Path, Restricted, α-Restricted

- ewaluacja All-subtrees, ze względu na uwzględnienie *wszystkich* klasyfikatorów potomnych, daje najbardziej wygładzone granice klasyfikacji; często jednak w obszarach gdzie sąsiadują duże dominujące obszary z małymi, te drugie mogą całkiem znikać,
- przeciwnie, Single-Path daje poszarpane granice obszarów: to efekt ewaluacji tylko jednego klasyfikatora potomnego, dając dobre wyniki klasyfikacji na samym zbiorze uczącym, jednak dużo słabsze wyniki generalizacji,
- ewaluacja typu Restricted daje wyniki zbliżone do All-subtrees, jednak nie pozwala na znikanie małych obszarów decyzji,
- ewaluacja α-Restricted jest podobna do Restricted.

3.2. Rozpoznawanie przedmiotów z bazy COIL i porównanie z innym modelem hierarchicznym

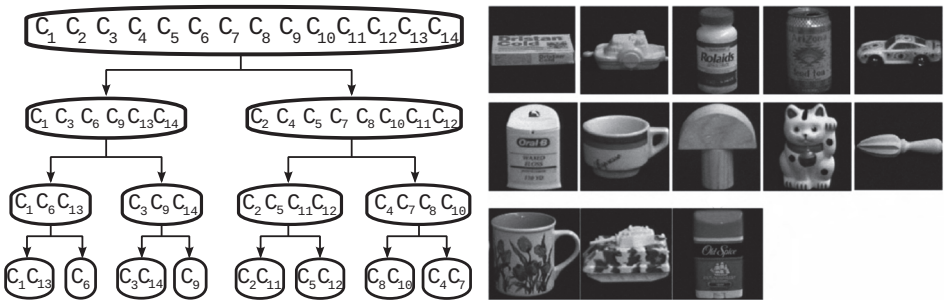
Wang, Casasent (2009) rozpatrują rozpoznawanie obrazów z bazy COIL (Nene *et al.*, 1996). Oryginalna baza zawiera kolorowe zdjęcia 100 obiektów o rozmiarach 128×128 pikseli. Każdy obiekt jest sfotografowany na czarnym tle z 72 pozycji co 5° stopni. Zdjęcia są przeskalowane tak, aby obiekt wypełniał je w całości. Wang, Casasent (2009) analizują rozpoznawanie 10 z tych obiektów w skali szarości zdjęć zredukowanych do rozmiaru 32×32 pikseli.

Model (Wang, Casasent, 2009) został już pokrótce opisany w rozdziale jako podobny do HCOC. Jak widać na rysunku 3.4, na każdym etapie klasyfikator w węźle dzieli przestrzeń klas na rozdzielne podzbiory klas (ang. *macro-class*). Zadanie to jest wykonywane przez zmodyfikowany model maszyny SVM, który poprzez klastrowanie w przestrzeni cech wyszukuje zawsze 2 klastry, generując dla każdego maszynę SVM odpowiadającą modelowi *grupa klas-przeciwko-wszystkim innym*. W trakcie nauczania autorzy podają po 24 obrazy (co 15°) dla 10 obiektów (górne 10 obiektów na rysunku 3.4). Dla testowania użyte jest po 8 zdjęć każdego z obiektów, różnych od tych w zbiorze uczącym. Jednocześnie autorzy testują zdolność modelu do odrzucania obiektów z klas, które nie pochodzą z nauczanych klas. W zbiorze testowym odrzucania wykorzystane jest po 8 zdjęć trzech obiektów przedstawionych u dołu rysunku 3.4.

Ze względu na inny typ klasyfikatorów bazowych nie jest możliwe dokładne powtórzenie eksperymentu. Model (Wang, Casasent, 2009) wykorzystuje modyfikacje SVM, natomiast HCOC, z założenia, używa w węzłach prostych modeli typu warstwowych sieci neuronowych z jedną warstwą ukrytą z niewielką liczbą neuronów. W związku z tym inna była reprezentacja obrazów: każdy obiekt był przedstawiany jako sklejenie histogramów kilku filtrów: jasności, krawędzi oraz Laplacian of Gausians. Ta reprezentacja była i tak nadmiarowa. Wszystkie histogramy miały po 32 elementy, jednak wyniki dla histogramów złożonych z 16 elementów były porównywalne.

Jednocześnie model HCOC nie jest przewidziany dla zadania odrzucania niewidzianych wcześniej klas. Odrzucanie w modelu Wanga i Casasenta ma miejsce wtedy, gdy końcowe aktywacje dla wszystkich rozpoznawanych klas osiągają wartości poniżej pewnego progu. W HCOC, w trakcie ewaluacji, wektory aktywacji są normalizowane, wobec tego zawsze któraś z aktywacji będzie osiągać wartość powyżej pewnego logicznego progu, np. prawdopodobieństwa *a priori* klasy $P(C_i)$. Ze względu na specyfikę budowy węzłów taka sytuacja jest możliwa w modelach Wanga i Casasenta.

Wyniki porównania dla problemu COIL pokazane są w tabeli 3.1. Dla modelu Wanga i Casasenta uwzględnione są wyniki pozwalające na 10% niepoprawnego odrzucania niepoprawnych klas. Jak widać, wyniki są porównywalne.



Rysunek 3.4. Model hierarchiczny Wanga i Casasenta. Po prawej wykorzystywane w nauczaniu obiekty: u góry obiekty rozpoznawane, u dołu nierozpoznawane, które model powinien odrzucić

Tabela 3.1. Porównanie wyników HCOC z tymi dla modelu Wanga i Casasenta

Model	Skuteczność (%)
Wang i Casasent: Hierarchiczny SVRDM miękki podział	100
Wang i Casasent: Hierarchiczny SVRDM twardy podział	100
Wang i Casasent: Hierarchiczny SVM miękki podział	99
HCOC: 3 neurony ukryte, `fitness.E`	100
HCOC: 3 neurony ukryte, `fitness.G`	98

3.3. Zbiory porównawcze z repozytoriów

Tabela 3.2 pokazuje skuteczności HCOC dla szeregu problemów testowych z bazy UCI (Newman *et al.*, 1998; Zwitter, Sokolic, 1988). W tabeli 3.3 podane są wyniki w porównaniu z uzyskanymi w znanej pracy (Setiono, 2001), w której autor postawił sobie zadanie znalezienia optymalnej liczby neuronów ukrytych przy wykorzystaniu algorytmu wykorzystującego walidację krzyżową. Setiono, dla oszacowania błędów, wykorzystywał 10-krotną walidację krzyżową. W wynikach dla HCOC wykorzystana jest miara błędu $Err^{(.632)}$ będąca rozszerzeniem walidacji krzyżowej (Efron, 1983, patrz dodatek A.13).

3.4. Klasyfikacja tekstur

W tym zadaniu celem była klasyfikacja tekstur z bazy University of Southern California (Brodatz, 1966). Z każdej tekstury losowane było po kilkanaście mniejszych obszarów, a każdy z nich reprezentowany jako 32-elementowy histogram jasno-

Tabela 3.2. Wyniki HCOC dla zbiorów z repozytorium UCI. Ewaluacja korzenia, korzenia i jednego, dwóch, trzech poziomów; *hid* to liczba ukrytych neuronów

Problem (klasy, ewaluacja)	*hid*	Błąd i odchylenie dla modelu HCOC			
		Błąd testowy dla HCOC na poziomie (%)			
		0	1	2	3
Audiology (24, α-Restr)	10	22.09 ± 4.6	6.56 ± 0.7	6.16 ± 0.8	6.19 ± 0.8
Audiology (24, Restr)	15	10.51 ± 1.8	6.04 ± 0.8	5.80 ± 0.8	6.01 ± 0.8
Audiology (24, Restr)	20	8.75 ± 1.0	5.64 ± 0.6	5.82 ± 0.7	5.93 ± 0.4
Cardiotography (10, α-Restr)	15	45.52 ± 7.3	42.69 ± 4.6	33.83 ± 3.4	36.75 ± 2.8
Cardiotography (10, Restr)	15	45.52 ± 7.3	42.79 ± 4.6	34.06 ± 3.5	37.42 ± 3.4
Cardiotography (10, α-Restr)	20	43.95 ± 6.7	39.75 ± 3.2	35.05 ± 2.0	37.23 ± 3.2
Cardiotography (10, all)	20	43.95 ± 6.7	39.77 ± 3.5	35.68 ± 2.0	37.98 ± 3.4
Libras (15, α-Restr)	3	67.34 ± 2.8	36.78 ± 1.5	26.35 ± 2.1	22.05 ± 1.2
Libras (15, α-Restr)	5	52.60 ± 4.4	25.23 ± 1.1	20.01 ± 1.4	19.42 ± 2.1
Libras (15, α-Restr)	7	39.63 ± 4.7	21.77 ± 1.8	18.39 ± 1.2	18.03 ± 1.4
Libras (15, Restr)	10	30.26 ± 2.2	17.54 ± 1.6	17.76 ± 1.6	17.13 ± 1.2
Libras (15, α-Restr)	10	32.46 ± 1.9	17.75 ± 1.8	17.46 ± 1.1	17.29 ± 1.2
Libras (15, α-Restr)	15	23.84 ± 1.9	16.40 ± 1.5	16.24 ± 1.5	15.35 ± 1.0
Primary tumor (21, α-Restr)	10	54.07 ± 1.7	46.43 ± 1.2	48.21 ± 1.6	49.23 ± 1.7
Primary tumor (21, Restr)	10	54.38 ± 2.3	46.71 ± 1.4	46.81 ± 1.3	48.09 ± 1.5
Primary tumor (21, α-Restr)	20	48.91 ± 1.4	44.63 ± 1.6	45.48 ± 1.6	45.82 ± 1.8
Primary tumor (21, Restr)	20	49.77 ± 1.4	44.29 ± 1.7	44.94 ± 0.8	45.20 ± 1.3
Segment (7, α-Restr)	3	10.57 ± 2.8	4.20 ± 0.6	3.60 ± 0.5	3.81 ± 0.6
Segment (7, all)	5	4.12 ± 0.7	3.48 ± 0.3	3.20 ± 0.2	3.27 ± 0.2
Segment (7, α-Restr)	5	4.17 ± 0.7	3.35 ± 0.2	3.25 ± 0.3	3.20 ± 0.4
Segment (7, α-Restr)	7	3.96 ± 0.5	3.18 ± 0.3	3.09 ± 0.3	3.05 ± 0.4
Segment (7, α-Restr)	10	4.13 ± 0.3	3.16 ± 0.3	3.02 ± 0.3	3.03 ± 0.2
Segment (7, Restricted)	15	3.31 ± 0.3	2.90 ± 0.2	2.95 ± 0.2	2.88 ± 0.2
Soybean (19, α-Restr)	2	58.52 ± 6.9	27.28 ± 3.0	18.00 ± 2.9	16.12 ± 2.5
Soybean (19, all)	5	23.27 ± 2.8	8.26 ± 0.9	7.39 ± 0.7	7.43 ± 0.7
Soybean (19, α-Restr)	7	16.54 ± 2.4	7.51 ± 0.8	7.20 ± 0.9	6.92 ± 0.9
Soybean (19, Restricted)	10	12.04 ± 0.9	7.00 ± 0.7	6.98 ± 0.5	6.82 ± 0.4
Soybean (19, α-Restr)	10	12.04 ± 0.8	7.01 ± 0.7	6.99 ± 0.5	6.85 ± 0.4
Soybean (19, Restricted)	15	10.62 ± 1.4	6.77 ± 0.6	6.56 ± 0.5	6.45 ± 0.6
Vowel (11, Restr)	5	26.40 ± 1.7	15.45 ± 0.9	14.01 ± 0.6	13.15 ± 0.8
Vowel (11, α-Restr)	5	26.40 ± 1.7	15.48 ± 1.0	14.11 ± 0.6	13.19 ± 0.8
Vowel (11, all)	5	25.63 ± 1.7	15.37 ± 0.6	13.46 ± 1.0	13.77 ± 0.8
Vowel (11, Restr)	10	18.20 ± 1.7	12.52 ± 1.4	11.96 ± 0.7	11.67 ± 1.1
Vowel (11, α-Restr)	10	19.75 ± 2.5	13.14 ± 0.9	12.23 ± 0.6	11.57 ± 1.0
Vowel (11, all)	10	21.42 ± 1.7	13.62 ± 1.4	12.72 ± 1.5	12.60 ± 1.1
Zoo (7, Restr)	5	4.65 ± 3.3	2.32 ± 1.4	2.17 ± 1.1	1.99 ± 0.8
Zoo (7, α-Restr)	5	4.65 ± 3.3	2.32 ± 1.5	2.17 ± 1.1	1.99 ± 0.8
Zoo (7, all)	5	4.65 ± 3.3	2.32 ± 1.5	2.17 ± 1.1	2.08 ± 0.9
Zoo (7, all)	5	3.53 ± 0.9	2.45 ± 0.9	2.53 ± 1.5	1.78 ± 0.9
Zoo (7, α-Restr)	10	1.76 ± 1.0	1.68 ± 0.6	1.59 ± 0.9	1.34 ± 0.9

Tabela 3.3. Wyniki porównania skuteczności działania modelu HCOC dla różnych HCOC z 0 (sam korzeń), 1, 2, 3 poziomami. Wyniki porównane z tymi w pracy (Setiono, 2001), gdzie zadaniem było znalezienie optymalnych architektur dla zadanych problemów

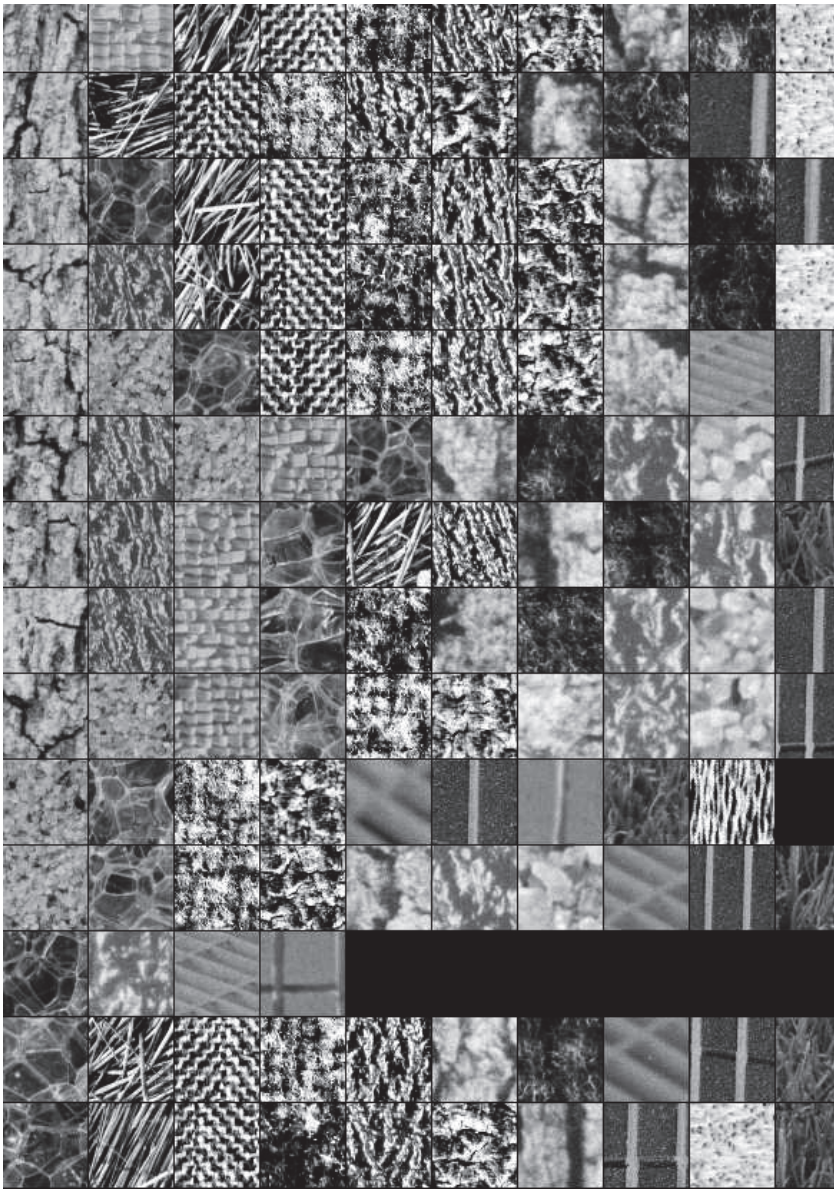
Problem (klasy, ewaluacja)	Dokładność HCOC (%)					Setiono (2001)	
	hid	$Err^{(0.632)}$ dla poziomu				*hid*	test err.
		0	1	2	3		(%)
Vowel (11, α-Restr)	3	68.92	46.47	37.04	27.23	17.4 ± 1.4	11.14 ± 1.5
Vowel (11, α-Restr)	5	47.87	25.41	20.10	16.86	17.4 ± 1.4	11.14 ± 1.5
Vowel (11, α-Restr)	7	28.29	15.95	14.92	13.02	17.4 ± 1.4	11.14 ± 1.5
Tumour (21, α-Restr)	4	61.81	53.25	50.44	49.44	11.4 ± 1.4	54.03 ± 1.4
Audiology (24, Restr.)	3	62.02	40.44	33.84	28.40	17.2 ± 1.2	20.50 ± 1.6
Audiology (24, α-Restr)	5	49.43	30.09	25.06	24.67	17.2 ± 1.2	20.50 ± 1.6
Soybean (19, α-Restr)	5	44.63	16.47	12.10	10.45	19.4 ± 0.7	7.03 ± 0.7
Soybean (19, α-Restr)	7	18.86	5.99	5.39	5.38	19.4 ± 0.7	7.03 ± 0.7
Zoo (7, Restr.)	2	26.26	13.39	10.34	8.78	12.1 ± 0.1	5.66 ± 1.5
Zoo (7, α-Restr)	5	10.61	6.70	5.79	5.79	12.1 ± 0.1	5.66 ± 1.5
Zoo (7, Restr.)	10	5.22	5.09	4.94	4.67	12.1 ± 0.1	5.66 ± 1.5

ści. W ten sposób zbiór uczący składał się z kilkuset przykładów uczących. Na rysunku 3.5 pokazany jest podział 20 różnych tekstur na klastry.

Nauczanie HCOC z 3-poziomami, wykorzystując *wyłącznie* histogram jasności, daje 94% poprawności na zbiorze uczącym i 86% poprawności na zbiorze testującym (obydwa wyniki pochodzą z walidacji krzyżowej) na zbiorze 32 tekstur. Dla porównania w tabeli 3.4 przedstawiono wyniki wykorzystujące dla reprezentacji różne kombinacje macierzy GLCM (ang. *Grey Level Co-occurence Matrices*) składających się z 5 do 14 filtrów, w szczególności filtrów energii, entropii, kontrastu, homogeniczności obrazu oraz szeroko wykorzystywanego filtru Gabora. Wyniki podano na podstawie pracy (Tou *et al.*, 2007) z późniejszymi uaktualnieniami wyników autora.

3.5. Zastosowania w teorii automatów

Ważnym zagadnieniem w projektowaniu automatów jest problem, czy dany automat jest synchronizujący. Automat $\mathcal{A}$ jest synchronizujący, jeśli istnieje dla niego słowo W, które przeprowadza automat do jednego stanu bez względu na stan



Rysunek 3.5. Podział na klastry w problemie rozpoznawania tekstur. Każdy wiersz reprezentuje klaster przedstawiony przez szereg tekstur w nim zawartych

początkowy (Kari, 2003). Ważnym zadaniem jest znalezienie najkrótszego słowa synchronizującego dany automat. Jednym z ostatnich nierozwiązanych zagad-

Tabela 3.4. Wyniki klasyfikacji tekstur na podstawie pracy (Tou *et al.*, 2007)

Model i użyte filtry	Skuteczność (%)
Tou: GLCM (5 filtrów)	86
Tou: Gabor	80
Tou: GLCM (5 filtrów) + Gabor	91
Tou: Macierz kowariancji (krawędzie)	85
Tou: Macierz kowariancji (GLCM (5 filtrów))	80
Tou: Macierz kowariancji (Gabor)	92
HCOC: histogram filtru jasności	86

nień informatyki jest tzw. hipoteza Ĉernego mówiąca, że dla n-stanowego automatu synchronizującego istnieje słowo synchronizujące o długości co najwyżej $(n-1)^2$ (Ĉerný *et al.*, 1971). Problem znalezienia minimalnego słowa synchronizującego (ang. *Minimal Synchronizing Word*, MSW) jest problemem trudnym (Olschewski, Ummels, 2010; Trahtman, 2006).

Można przedstawić problem wyszukiwania minimalnego słowa synchronizującego jako problem klasyfikacyjny, gdzie dla zadanego automatu klasyfikator zwraca długość MSW. Algorytmy wyszukujące minimalne słowa synchronizujące mają złożoność wykładniczą, wobec czego przydatne jest uproszczenie rozwiązania przez najpierw przewidzenie jego prawdopodobnej długości, a następnie skupienie się na węższym przedziale rozwiązań.

To zadanie składa się z dwóch problemów: z jednej strony, wykorzystania klasyfikatora hierarchicznego HCOC, z drugiej, odpowiedniej reprezentacji automatu. Wykorzystanie HCOC jest uzasadnione ze względu na jego przeznaczenie do rozwiązywania problemów z dużą liczbą klas wyjściowych: można się wobec tego spodziewać szybkiego wyekstrahowania z przedziału oczekiwanych długości MSW $[0,(n-1)^2]$ (jeżeli, oczywiście, hipoteza Ĉernego jest prawdziwa) wąskiego podprzedziału, dla którego można uruchomić algorytm wyszukujący konkretne słowo.

Klasyfikator HCOC jednak, podobnie jak inne modele, przyjmuje jedynie wektory atrybutów o ustalonej długości. Nie jest więc możliwe przedstawienie automatu jako tablicy przejść. Okazuje się jednak, że znacznie lepszym podejściem jest jego reprezentacja jako wektora wstępnie obliczonych cech (Podolak *et al.*, 2012a).

W doświadczeniu wykorzystane zostały następujące cechy

- $gcd()$ długości cykli dla każdej litery alfabetu z osobna: dla automatu na rysunku 3.6 i dla litery a ta wartość wynosi $F_1(a) = gcd(2,4) = 2$,
- średnia długość cyklu dla zadanej litery: $F_2(a) = (2+4)/2 = 3$,

- maksymalna długość ścieżki wchodzącej do cyklu dla danej litery: $F_3(a) = \max(|((q_3, q_2))|, |((q_6, q_9), (q_9, q_8))|, |((q_{10}, q_8))|) = 2$,
- ułamek liczby cykli, dla danej litery alfabetu, przypadających na jeden stan: $F_4(a) = (2 + 4)/10 = 0.6$,
- rozmiar synchronizowalnej części automatu dla danej litery alfabetu automatu: $F_5(a) = |\{q_2, q_8\}|$,
- średni stopień synchronizowalności automatu dla danej litery: $F_6(a) = (8 + 2 + 3)/10 = 1.3$,
- średnia wysokość stanów w automacie par: $F_7(\mathcal{A}) = 3.77$.

Wszystkie cechy są proste w obliczeniu.

W trakcie doświadczeń nauczono HCOC dla automatów z $|Q| = 4, 5, 6, 8$ stanami oraz $|A| = 2, 3, 4$ literami alfabetu. Dla każdego układu wygenerowane zostały tysiącelementowe zbiory uczące dla losowo wygenerowanych automatów, które zostały podzielone losowo na zbiory uczące i testujące. Zadanie zostało przedstawione jako problem klasyfikacji: nauczony model miał za zadanie wskazać klasę skojarzoną z właściwą długością MSW. Wyniki zostały porównane z tymi dla warstwowej sieci neuronowej (tabela 3.5). Tabela 3.6 pokazuje korelacje między wartościami cech a długością MSW.

HCOC znajduje prawie idealne długości MSW (patrz błąd diff err w tabeli 3.5), przy czym te predykcje są stabilne, dużo bardziej niż te dla sieci neuronowej. Niskie są wartości odchylenia kwadratowego. Zakładając prawdziwość hipotezy Ĉernego, ta długość jest kwadratowa ze względu na liczbę stanów, stąd będzie rosła szybko wraz ze wzrostem wielkości automatów i stabilność odpowiedzi będzie szalenie istotna. Trzeba zauważyć, że wypróbowany tu zbiór cech F_1 do F_7 nie jest zamknięty i można oczekiwać znacznie lepszych wyników dla nowo wyliczonych cech.

3.6. Rozpoznawanie twarzy

W problemach rozpoznawania twarzy zadaniem jest prawidłowa klasyfikacja obrazu i przypisanie go jednej ze znanych osób na podstawie obrazów (Podolak *et al.*, 2002; Podolak, Roman, 2011a). Problemami są zła rozdzielczość obrazów, różne oświetlenie i pozy itp.

W tym doświadczeniu wykorzystana została baza twarzy AT&T (Samaria, Harter, 1994), składająca się ze zdjęć 40 różnych osób, po 10 zdjęć każdej osoby. Każde zdjęcie ma 92×112 pikseli w 256 stopniach szarości. Zdjęcia zostały najpierw poddane analizie PCA i wybranych zostało 100 wektorów własnych odpowiadających 100 najwyższym wartościom własnym.

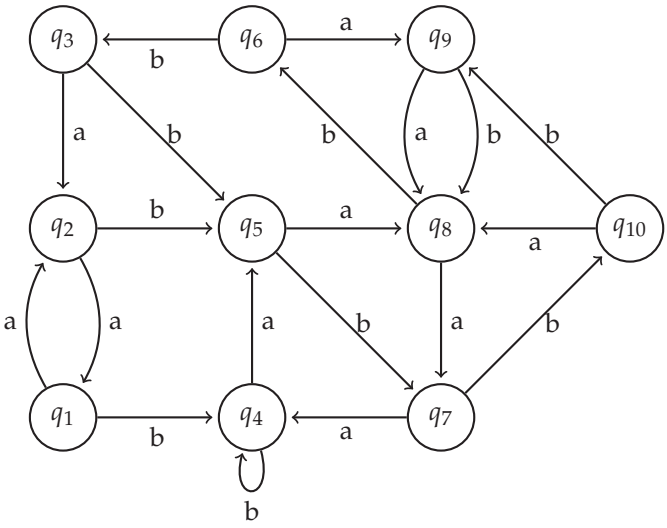
Tabela 3.5. Wyniki predykcji długości minimalnego słowa synchronizującego; *hid* to liczba ukrytych neuronów, *diff err* to błąd jako suma bezwzględnych różnic w odpowiedziach, *MSE* to średni błąd kwadratowy odpowiedzi. Dla każdej kolumny podana także wartość błędu dla najlepszego przebiegu (wypośrodkowana)

			Klasyfikator HCOC			Sieć neuronowa		
$\|Q\|$	$\|A\|$	*hid*	ℓ_{01}	*diff err*	*MSE*	ℓ_{01}	*diff err*	*MSE*
4	2	8	0.31 ± 0.03 0.25	0.42 ± 0.04 0.32	0.77 ± 0.14 0.50	0.71 ± 0.10 0.37	1.62 ± 0.60 0.43	5.19 ± 2.80 0.57
4	3	3	0.43 ± 0.06 0.34	0.60 ± 0.13 0.44	1.07 ± 0.40 0.71	0.66 ± 0.14 0.43	1.20 ± 0.38 0.59	2.82 ± 1.24 0.97
4	4	8	0.47 ± 0.04 0.40	0.61 ± 0.08 0.47	0.97 ± 0.22 0.62	0.71 ± 0.09 0.60	1.22 ± 0.27 0.77	2.4 ± 0.80 1.17
5	2	8	0.55 ± 0.03 0.50	0.89 ± 0.07 0.73	1.86 ± 0.25 1.37	0.82 ± 0.11 0.51	2.57 ± 0.90 0.70	11.4 ± 6.53 1.24
5	3	5	0.49 ± 0.05 0.44	0.71 ± 0.29 0.58	1.37 ± 1.63 0.94	0.70 ± 0.18 0.47	1.48 ± 0.72 0.64	4.27 ± 2.92 1.05
5	4	8	0.44 ± 0.02 0.40	0.60 ± 0.05 0.52	1.14 ± 0.16 0.94	0.74 ± 0.17 0.40	1.86 ± 0.72 0.53	6.84 ± 3.21 0.94
6	2	3	0.63 ± 0.05 0.58	1.11 ± 0.21 0.93	2.75 ± 1.00 1.97	0.85 ± 0.12 0.59	3.24 ± 1.31 0.88	18.8 ± 12.0 1.67
6	3	8	0.64 ± 0.01 0.61	1.11 ± 0.06 1.01	2.75 ± 0.35 2.36	0.82 ± 0.13 0.64	2.93 ± 1.54 0.97	17.2 ± 12.9 1.98
6	4	8	0.55 ± 0.05 0.51	0.80 ± 0.48 0.67	1.83 ± 4.26 1.07	0.82 ± 0.11 0.51	2.61 ± 1.06 0.69	13.6 ± 7.88 1.23
8	2	5	0.76 ± 0.02 0.73	1.90 ± 0.15 1.70	7.04 ± 1.08 5.67	0.86 ± 0.10 0.71	4.25 ± 2.66 1.61	38.2 ± 36.3 5.09
8	3	8	0.70 ± 0.02 0.67	1.44 ± 0.13 1.31	4.67 ± 0.91 3.79	0.88 ± 0.08 0.70	4.00 ± 1.58 1.28	29.7 ± 16.8 3.60
8	4	5	0.77 ± 0.01 0.74	2.20 ± 0.21 1.75	10.29 ± 2.05 6.41	0.91 ± 0.05 0.77	5.00 ± 1.72 1.71	42.4 ± 27.1 5.52

We wszystkich testach wykorzystano sieci neuronowe z siedmioma neuronami w korzeniu. Wszędzie wykorzystana została metoda All-subtrees dla ewaluacji. Rysunki 3.7 i 3.8 pokazują podział zadania na klastry poprzez wskazanie, która osoba była rozpoznawana przez który klaster (rysunki pokazują po jednej, wybranej losowo z bazy, twarzy dla każdej osoby). Rysunki 3.7 i 3.8 przedstawiają podziały twarzy na klastry w zbudowanych modelach.

Tabela 3.6. Korelacje między wartościami cech a długością minimalnego słowa synchronizującego

$\|Q\|$	$\|A\|$	F_1	F_2	F_3	F_4	F_5	F_6	F_7
4	2	0.23	0.33	−0.07	0.28	−0.15	−0.27	0.93
4	3	0.24	0.29	−0.10	0.30	−0.22	−0.33	0.90
4	4	0.18	0.26	−0.20	0.38	−0.26	−0.37	0.90
5	2	0.27	0.35	−0.15	0.38	−0.21	−0.33	0.90
5	3	0.18	0.34	−0.27	0.50	−0.32	−0.41	0.92
5	4	0.10	0.19	−0.27	0.44	−0.31	−0.39	0.93
6	2	0.19	0.35	−0.26	0.50	−0.28	−0.34	0.93
6	3	0.10	0.16	−0.32	0.50	−0.36	−0.40	0.94
6	4	0.12	0.20	−0.42	0.56	−0.48	−0.48	0.95
8	2	0.10	0.31	−0.40	0.63	−0.30	−0.32	0.94
8	3	0.04	0.11	−0.32	0.50	−0.30	−0.38	0.92
8	4	−0.07	−0.03	−0.35	0.48	−0.36	−0.40	0.90



Rysunek 3.6. Przykład automatu synchronizującego. Wektor cech dla tego automatu ma postać $(2, 3, 1, 0.6, 2, 1.3, 1, 4, 1, 0.8, 2, 1.2, 3.77)$

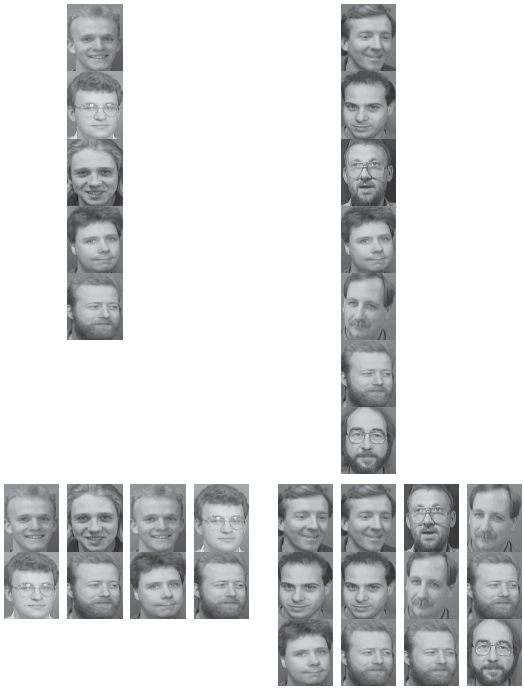
Błędy kwadratowe pojedynczego monolitycznego modelu (tu sieci warstwowej neuronowej), obliczony przy wykorzystaniu metody walidacji krzyżowej, wynosiły dla korzenia 0.48 dla zbioru trenującego i 0.69 dla testującego. Był on więc

stosunkowo wysoki, chociaż, pamiętając, że razem jest 40 klas w problemie, oczywiście znacznie lepszy niż dla klasyfikatora losowego. Jednak klasyfikator hierarchiczny, przy wykorzystaniu łącznie 3 poziomów, dawał błąd 0.20 dla zbioru trenującego i 0.38 dla testującego. Celem porównania było tu głównie pokazanie, że HCOC znacznie poprawia błąd. Przy lepszej reprezentacji, np. wykorzystując ekstrakcję cech twarzy przy wykorzystaniu innych algorytmów (Podolak *et al.*, 2002), należy się spodziewać lepszych rezultatów. Do nauczania użyta została implementacja HCOC w języku Java przy wykorzystaniu algorytmu GNG do klastrowania oraz algorytmu typu Resilient Propagation dla nauczania warstwowych sieci neuronowych w węzłach.

Rysunek 3.7. Część drzewa klastrów w zadaniu rozpoznawania twarzy. U góry klastry pierwszego poziomu. W nauczaniu każda z 40 twarzy była reprezentowana przez pięć z dziesięciu dostępnych zdjęć. Ostatni z klastrów pierwszego poziomu nie został już podzielony. Zdjęcia z bazy AT&T

3.7. Ekstrakcja reguł

Ciekawym problemem nauczania maszynowego jest ekstrakcja reguł z nauczonych modeli (Andrews *et al.*, 1995; Duch *et al.*, 2004b; Setiono, Leow, 2000), która może być zastosowana także do modeli hierarchicznych (Kurzyński, 1983). Modele takie jak sieci neuronowe uzyskują wysoki stopień generalizacji, jednak działają



Rysunek 3.8. Część drzewa klastrów z innego doświadczenia w zadaniu rozpoznawania twarzy. U góry klastry pierwszego poziomu. Zdjęcia z bazy AT&T (Samaria, Harter, 1994)

jak „czarna skrzynka", która pobiera dane wejściowe i zwraca wyniki, ale nie pokazuje *dlaczego* takie właśnie wyniki. W wielu systemach, np. biznesowych, bankowych, dotyczących ubezpieczeń, zwrócone odpowiedzi powinny nie tylko być prawidłowe, ale także podlegać kontroli człowieka i pozwalać na ich indywidualną akceptację[1]. Ma to szczególne znaczenie w okresie wprowadzania systemu do ciągłego użytku.

Możliwym podejściem jest próba ekstrakcji reguł typu IF-THEN z nauczonego modelu dającego dobrą generalizację. Dokładność takiego modelu jest oczywiście niższa niż modelu pierwszego ze względu na niedokładność samej ekstrakcji, jednak będzie pozwalać na analizę sposobu przeszukiwania wiedzy przez model pierwotny. Takie reguły są jednak w dalszym ciągu słabo czytelne. Na przykład w przypadku sieci neuronowych będą zwykle wykorzystywały aktywacje neuronów ukrytych, które same w sobie opisują cechy trudne do interpretacji.

[1]System ekspertowy MYCIN, służący do analizy bakteryjnych zakażeń krwi, mimo osiągnięcia „profesorskiego" poziomu poprawności analizy, nigdy nie wszedł do szerszego użytku. Stało sie tak w dużym stopniu ze względu na słabą wytłumaczalność odpowiedzi, a co za tym idzie, ograniczone zaufanie lekarzy.

Tabela 3.7. Skuteczność klasyfikatora HCOC w porównaniu ze skutecznością systemu ekspertowego z wykorzystaniem wydobytych reguł. HCOC z sieciami neuronowymi jako klasyfikatorami bazowymi o różnej liczbie neuronów ukrytych. Wyniki w %

	1 ukryty		2 ukryte		3 ukryte		4 ukryte		5 ukrytych	
	HC	reguły	HC	reguły	HC	reguły	HC	reguły	HC	reguły
Audiology	59.3	30.9	85.6	47.2	92.6	44.2	95.0	71.5	95.7	84.4
Primary tumor	35.1	24.8	53.7	37.5	65.6	44.9	70.0	47.2	73.3	48.8
Vowel	44.4	13.7	88.8	67.7	95.3	84.6	97.0	88.1	97.9	90.5
Zoo	82.8	63.0	93.4	86.8	99.4	96.9	99.7	99.5	99.7	99.7

Możliwym częściowym rozwiązaniem jest próba ekstrakcji reguł z nauczonego klasyfikatora HCOC (Podolak, 2007). W systemie wykorzystano klasyfikator HCOC z sieciami neuronowymi jako klasyfikatorami bazowymi Cl. Dla każdego klasyfikatora bazowego Cl zastosowany został algorytm ekstrakcji Fernn (Setiono, Leow, 2000), generując reguły typu

$$\text{If } Predicate(x) \text{ Then } Class(x) = [C_1, \ldots, C_K], \tag{3.1}$$

gdzie x jest wektorem wejściowych atrybutów, $Predicate(x)$ pewnym logicznym predykatem, $Class(x) = [C_1, \ldots, C_K]$ jest wektorem aktywacji. Wykorzystując zadany sposób ewaluacji drzewa (najlepiej Restricted lub α-Restricted dające ściśle określone progi „rozpoznania" klasy), algorytm ekstrakcji szuka wartości predykatu $Recognized(Q^i)$ oznaczającego rozpoznanie klasy należącej do klastra Q^i. Wtedy możliwe jest wydobycie kolejnych reguł

$$\text{If } Recognized(Q^i) \wedge PredictedCluster_k^{Cl^i}(x) \text{ Then } Class(x) = [C_1, \ldots, C_K], \tag{3.2}$$

gdzie $PredictedCluster_k^{Cl^i}(x)$ jest pewnym predykatem o wartości obliczonej przez klasyfikatora Cl^i. Analogiczna reguła

$$\text{If } Recognized(Q^i) \wedge PredictedCluster_k^{Cl^i}(x) \text{ Then } Cluster(x) = [w_1, \ldots, w_L], \tag{3.3}$$

gdzie $[w_1, \ldots, w_L]$ oznacza wektor wag klastrów, rozpoznaje klastry.

W trakcie doświadczeń utworzono taki zbiór reguł dla wielu ogólnie znanych zbiorów wykorzystywanych w problemach nauczania maszynowego (Newman *et al.*, 1998). Wygenerowane reguły zostały wpisane do systemu ekspertowego typu Rete (Friedmann-Hill, 2003) dla porównania. Wyniki podane są w tabeli 3.7.

Klasyfikatory bazowe (sieci neuronowe z jedną warstwą ukrytą) miały redukowane w niewielkim stopniu połączenia dla redukcji liczby parametrów, a stąd

i reguł. Przykładowe klasyfikatory zostały zbudowane dla małej liczby ukrytych neuronów. Z wyników widać, że dla prostszych problemów wyniki systemu ekspertowego zaczynają się zbliżać do tych dla HCOC, natomiast dla problemów bardziej skomplikowanych jak primary tumor odstają one w znacznym stopniu. Przykładowa reguła wydobyta z systemu dla problemu zoo (problem z bazy UCI), w którym szereg zwierząt opisanych jest za pomocą wektora cech takich, jak *czy ma płuca?*, *czy ma ogon?*, *ile ma nóg?*, *jakiej jest wielkości?*, *jakie środowisko zamieszkuje?* i innych, a zadaniem systemu jest rozpoznanie rodziny, do której dane zwierzę należy, może wyglądać następująco:

```
if(3*"jaja"-2*"mleko"-1*"nogi"+4*"ogon" < 1.7) {
  if(-2*"pióra"+4*"jaja"-2*"mleko"+3*"wodne"-1*"wielkości kota" < 2.9) {
    Q={"ssak", "zwierzę wodne"}
  } else {
    if( ... ) {
      Q = {"ryba", "insekt"}
    } else {
      Q = {"gad", "mięczak"}
    }
  }
} else {
  ...
}
```

Wyrażenia są wciąż złożone, ponieważ pochodzą z analizy sieci neuronowych, jednak ze względu na to, że ich docelowymi klasami nie są pojedyncze zwierzęta, lecz grupy (klastry) zwierząt, stają się znacznie bardziej zrozumiałe i możliwe do analizy. Jednocześnie klastry przejmują rolę cech ukrytych występujących w sieciach neuronowych, jednak o czytelnym znaczeniu. Mogą one być przy tym zmodyfikowane, jeśli znana jest jakaś wiedza ekspercka na temat problemu, np. do systemu ekspertowego mogą być dodane nowe reguły operujące na tych samych predykatach.

Zakończenie

W pracy opisana została propozycja hierarchicznego modelu klasyfikacji. Model Hierarchicznego Klasyfikatora HCOC wyróżnia się wykorzystaniem nowego paradygmatu podziału oryginalnego problemu na podproblemy. Polega on na podziale w przestrzeni klas, nie przestrzeni atrybutów.

Nie jest to przy tym zwykły podział przestrzeni klas według ewentualnie dostępnej wiedzy eksperckiej czy też podział losowy. Polega na nauczeniu prostego klasyfikatora dla całego problemu i poprzez analizę znalezionego rozwiązania algorytm HCOC wyszukuje, które klasy są do siebie podobne i które klasy były przez oryginalny klasyfikator z sobą mylone. Algorytm HCOC znajduje prototypy aktywacji w przestrzeni rozwiązań i wokół nich buduje nowe podproblemy, które są rozwiązywane na kolejnych poziomach. Bardzo istotnym elementem rozwiązania jest fakt, że budowane podproblemy przecinają się. Powoduje to, że pojedyncze klasy mogą należeć do więcej niż jednego podproblemu, co polepsza skuteczność algorytmu.

Algorytm znajduje więc nową wiedzę poprzez analizę, co ten pierwszy klasyfikator ‚myśli", co mu się „wydaje" o problemie. Pomimo tego że klasyfikator w korzeniu poddrzewa myli się, wydobyta przez niego wiedza jest wydobyta i wykorzystana. Analiza słabych klasyfikatorów, umieszczona w pracy, pozwala na przyjęcie, iż słabo nauczony pierwszy klasyfikator bazowy *nie* daje odpowiedzi losowych, że wiedza wynikająca z takiej analizy ma sens.

W pracy zdefiniowane jest na nowo pojęcie *słabego* (*weak*) klasyfikatora, które lepiej, niż dotychczas znane w literaturze definicje, oddaje jego istotę. Nowa definicja bierze pod uwagę fakt, że każdy klasyfikator składa się z dwóch elementów: pierwszy oblicza wektor przynależności do klas, a drugi element wybiera spośród nich tę klasę, która wydaje się najbardziej prawdopodobna. Nowa definicja stara się uwzględnić właśnie prawdopodobieństwo właściwego wyboru i podaje warunki względem wartości aktywacji, aby wybrana została właśnie poprawna klasa. Przedstawione jest, jak nowa definicja wpływa na nauczanie klasyfikatora przy zmienionych warunkach.

W kolejnych rozdziałach pokazano, że klasyfikator HCOC minimalizuje swój błąd wraz z dodawaniem nowych warstw i przy zachowaniu pewnych warunków. Wskazują one pewne kierunki, które powinny być zachowane w trakcie nauczania. Sama konstrukcja klasyfikatora korzysta z tych rozwiązań teoretycznych.

W pracy przedstawiono wiele rozwiązań poszczególnych problemów, zaproponowane są nowe i zaadaptowane istniejące algorytmy oraz przeprowadzona ich analiza. W szczególności ważne są podane algorytmy agregacji wyników (SP, All-subtrees, Restricted, α-Restricted), które starają się coraz bardziej redukować nieuchronny szum i uniezależniać klasyfikator od niego. Pokazane jest, że HCOC wykorzystuje mechanizm agregacji pośrednio zależny od danych, który według innych badań jest najbardziej wydajny. Innym ważnym elementem pracy jest zdefiniowanie szeregu algorytmów rozdzielania klas na klastry tak, aby maksymalizować efekt rozdzielania obliczeń. Obliczenia mogą być wykonywane równolegle.

Model HCOC jest szkieletem klasyfikatora w postaci drzewa, w którym w każdym węźle wykorzystywany jest słaby klasyfikator. W implementacji pokazanej w pracy stosowane są modele sieci neuronowych oraz zdefiniowanego, na potrzeby HCOC, lasu drzew decyzyjnych. Możliwe są także inne klasyfikatory, jak chociażby pozostający w trakcie rozwoju liniowy klasyfikator losowy. Ważne jest to, że model HCOC nie jest architekturą zamkniętą, lecz poprzez wykorzystywanie różnych elementów składowych może być dostosowywany do konkretnych zapotrzebowań.

Ważną cechą HCOC jest jego szczególna przydatność do rozwiązywania problemów o bardzo dużej liczbie klas. Zaproponowana architektura może pozwolić na skalowalność rozwiązania ze względu na zwiększającą się wymiarowość wyjścia. To problem ważny obecnie.

Przyszłość HCOC leży w lepszym jeszcze zdefiniowaniu bazowych klasyfikatorów w węzłach. W dotąd zaimplementowanej architekturze klasyfikatory kolejnego poziomu nie wykorzystują wystarczająco, poza samym podziałem na podproblemy, wyników nauczania klasyfikatorów w korzeniu. Jeśli klasyfikatorami bazowymi są sieci neuronowe, to wynikami są także znalezione wektory wag i cechy wykrywane w warstwie ukrytej. Te cechy mogą być użyte na kolejnym poziomie. Gdy tak się stanie, HCOC stanie się pewnym przybliżeniem modelu głębokiego nauczania (ang. *deep learning*), który stanowi prawdopodobnie przyszłość sieci neuronowych (Hinton, Osindero, 2006; Bengio, LeCun, 2007). Praca nad takim rozwinięciem jest w toku.

Dodatek A

Problemy nauczania maszynowego

A.1. Nadzorowane nauczanie maszynowe

Każdy, w przypadku nauczania nadzorowanego, przykład opisany jest przez parę składającą się z *wektora atrybutów* oraz *etykiety klasy*. Na tej podstawie przeprowadzane jest nauczanie.

Definicja A.1. Dla ustalonego skończonego zbioru przykładów $D = \{(x_i, c_i)\}_{i=1}^{N}$, zwanego zbiorem uczącym, składającego się z N par p-elementowych wektorów atrybutów $x_i \in \mathcal{X}$ etykietowanych prawdziwą klasą $c_i \in \mathcal{C}$, algorytm klasyfikujący ma za zadanie znaleźć hipotezę $h : \mathcal{X} \longrightarrow \mathcal{C}$, która będzie tak zgodna z tymi etykietowaniami, jak to możliwe.

Poszczególne atrybuty mogą przyjmować wartości różnego typu, w szczególności mogą one być *ilościowe* (ang. *quantitative*) lub *jakościowe* (ang. *qualitative*).

A.2. Atrybuty przykładów uczących

Atrybuty przykładów (zmienne *niezależne*) opisujące klasyfikowane przykłady mogą być reprezentowane w różny sposób, w zależności zarówno od informacji, którą przekazują, jak i oczekiwanego sposobu klasyfikacji. Typy i sposoby reprezentacji atrybutów pokazane są w tabeli A.1. Podstawowe rozróżnienie między atrybutami ilościowymi i jakościowymi, gdy mówimy o typie zmiennych zależnych, definiuje także rozróżnienie między problemami regresji i klasyfikacji.

Tabela A.1. Różne typy i możliwości reprezentacji atrybutów

Atrybuty	Opis wykorzystania
ilościowe (ang. *quantitative*)	pomiary, które pozostają w relacji większości z sobą i istnieje miara określająca odległość między nimi
jakościowe (ang. *qualitative*) albo kategoryczne	opisują występowanie bądź nie jakichś cech; przy czym brak jest pomiędzy różnymi wartościami atrybutu żadnego znanego *explicite* porządku
jakościowe uporządkowane (ang. *ordered*)	wartości należą do zbioru etykiet, w którym można wyróżnić porządek, nie jest jednak znana *a priori* żadna metryka pozwalająca na określenie odległości pomiędzy tymi wartościami

A.3. Funkcje kosztu i ryzyka

Jeśli dostępny jest skończony zbiór przykładów, to może on stanowić *zbiór uczący* $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^{N}$ dla algorytmów nauczania maszynowego. Etykieta t_i jest prawdziwą (ang. *target*) klasyfikacją wektora obserwowalnych atrybutów x_i.

Zadaniem algorytmu ML jest zbudowanie modelu $h()$

$$T = h(X) + \epsilon, \tag{A.1}$$

gdzie ϵ jest losowym błędem (szumem) o wartości oczekiwanej $E[\epsilon] = 0$. W tym modelu hipoteza $h(x) = E[T|X = x]$ modeluje rozkład warunkowy $P(T|X)$. Warunek $E[\epsilon] = 0$ odpowiada oczekiwaniu, że hipoteza będzie podawać poprawną odpowiedź dla wielu wektorów atrybutów, również tych, które nie były znane na etapie nauczania. Konieczne jest tu określenie sposobu karania algorytmu za nieprawidłową predykcję: spoczywa ono na funkcji *kosztu* (ang. *loss function*).

Definicja A.2. Funkcja kosztu $\ell(x, t, h(x))$ jest pewną nieujemną funkcją $\ell : \mathcal{X} \times \mathcal{X} \times \mathcal{C} \longrightarrow \mathbb{R}_+ \cup \{0\}$ taką, że $\ell(x, t, h(x)) = 0$ gdy $h(x) = t$.

Definicja A.3. Funkcja ryzyka (ang. *risk function*) jest wartością oczekiwaną na przestrzeni przykładów testowych

$$R[h] = \int \ell(x, t, h(x)) P(x, t). \tag{A.2}$$

Rozkład prawdopodobieństwa $P(x, t)$ w definicji A.3 nie jest możliwy do znalezienia ze względu na dostępność, zwykle, jedynie skończonej liczby dostępnych przykładów. W takiej sytuacji musimy się zadowolić średnią wartością na zbiorze

Tabela A.2. Niektóre możliwe funkcje kosztu $\ell(x, t, h(x))$ dla problemów regresji i klasyfikacji do dwóch klas. Pierwsze trzy używane są w problemach regresji, pozostałe w klasyfikacji. Dla problemów wieloklasowych funkcje kosztu są pewnym złożeniem definicji dla problemów dwuklasowych z uwzględnieniem macierzy kosztu błędnych klasyfikacji między każdą parą różnych klas

Funkcja	Opis
$\|t - h(x)\|$	wartość bezwzględna
$(t - h(x))^2$	kwadratowa
$(t - h(x))^{2p}$	dla całkowitego $p > 1$
$I(t \neq h(x))$	0 – 1 (binarna)
$-2\sum_{k=1}^{K} h_k(x) log\, h_k(x)$	entropii wykorzystywana w modelu klasyfikacji
$\max(0, 1 - th(x))$	*soft margin*
$\ln(1 + \exp(-th(x)))$	logistyczna, w modelu Support Vector Machine (SVM)
$\exp(-th(x))$	wykładnicza, także w modelu SVM

przykładów dostępnych

$$R_{emp}[h] = \frac{1}{N}\sum_{i=1}^{N} \ell(x_i, t_i, h(x_i)), \tag{A.3}$$

zwaną *empiryczną* funkcją ryzyka (Scholkopf, Smola, 2002).

A.4. Klasyfikatory „monolityczne”

Większość problemów jest rozwiązywana zwykle przy wykorzystaniu modeli o jednolitej, „monolitycznej”, architekturze, które stosują jeden ustalony algorytm nauczania i budują jednolity system. Wśród wielu modeli najlepiej znanymi klasami są:

perceptron jest przykładem zastosowania modelu *dyskryminatywnego*, w którym model, mając dane odwzorowania wektorów atrybutów x i klasyfikacji c_i (np. w postaci zbioru uczącego), stara się bezpośrednio znaleźć granice podziałów (ang. *decision boundaries*). Perceptron jest bardzo uproszczonym modelem neuronu biologicznego. Ze względu na swoją budowę perceptron jest w stanie rozwiązywać całkowicie poprawnie jedynie zadania, które są *liniowo separowalne* (Rosenblatt, 1962; Minsky, Papert, 1969; Bishop, 1995);

perceptron wielowarstwowy jest rozszerzeniem modelu perceptronu przez dodanie, jednej lub więcej, warstw neuronów pomiędzy warstwę wejściową a wyjściową. Model jest tworzony przez znalezienie optymalnego zestawu

wag dla zdefiniowanej funkcji kosztu. Perceptron wielowarstwowy jest modelem nieparametrycznym, tworzonym przez przykłady (m.in. Rumelhart *et al.*, 1986; Haykin, 2009; Bishop, 2006).

Twierdzenie o aproksymacji mówi, że wykorzystując poprawnie zbudowany perceptron z jedną warstwą ukrytą, można znaleźć aproksymację dowolnej nieznanej funkcji celu z dowolnie małą dokładnością. Jednak jest to twierdzenie o charakterze egzystencjalnym i nie mówi, jak taki cel osiągnąć;

liniowa analiza dyskryminacyjna (ang. *Linear Dominant Analysis*, LDA) znajduje liniową kombinację atrybutów wejściowych, która odpowiada hiperpłaszczyźnie skutecznie rozdzielającej przykłady z różnych klas. LDA jest w tym sensie bliska modelowi perceptronu. Najbardziej znany jest model dyskryminacyjny Fishera (Fisher, 1936). Uogólnieniem LDA są metody, w których hiperpłaszczyzna rozdzielająca jest wyższego stopnia, np. kwadratowa jak w QDA (Vapnik, 1996);

drzewa decyzyjne są budowane przez rekurencyjny podział przestrzeni wejściowej wzdłuż wszystkich atrybutów, tworząc za każdym razem nowe węzły z mniejszą liczbą przykładów spełniających warunki. Są dzięki temu sekwencyjnemu podziałowi proste w budowie oraz w zrozumieniu podawanych predykcji.

Podział może być wykonywany aż do uzyskania drzewa z liśćmi, w których wszystkie przykłady należą do jednej klasy. Takie drzewa będzie jednak bardzo źle generalizować, stąd zwykle stosowane są procedury obcinania gałęzi (ang. *pruning*) rozsądnie powiększające błąd na zbiorze uczącym, jednak poprawiające generalizację. Jest wiele znanych algorytmów budowania drzew decyzyjnych, różniących się, przede wszystkim, wykorzystanym paradygmatem podziału, jak CART, ID3, C4.5, C5.0 i inne (np. Breiman *et al.*, 1984; Quinlan, 1993, 1996; Witten, Frank, 2000; Kohavi, Quinlan, 2002; Sethi, Sarvarayudu, 1982);

metody najbliższych sąsiadów dla zaklasyfikowania danego przykładu tworzą kulę obejmującą K najbliższych przykładów ze zbioru uczącego i klasyfikują zgodnie z przeważającą etykietą. Tu podstawowym problemem jest, tak zwane, *przekleństwo wymiarowości* (ang. *curse of dimensionality*);

samorganizujące się mapy (ang. *Self-Organizing Maps*, SOM) są modelem bardzo bliskim klastrowaniu. Przykłady z przestrzeni wejściowej są klastrowane w przestrzeni wyjściowej z narzuconą topografią siatki (np., dla interpretowalności, dwuwymiarowej).

Modele typu SOM pozwalają na redukcję wymiarowości przez połączenie atrybutów i usunięcie tych, które są zbędne. Wprowadzenie innych miar podobieństwa pozwala na grupowanie według zupełnie całkiem innych zasad (m.in. Kohonen, Honkela, 2007; Fritzke, 1993; Brugger *et al.*, 2008);

naiwny klasyfikator Bayesa zakłada, że wszystkie atrybuty są liniowo niezależne i, korzystając z prawa Bayesa, maksymalizuje przynależność przykładu do jednej z klas $P(C_k|x) \propto P(x|C_k)P(C_k)$.

Naiwny klasyfikator Bayesa oryginalnie wymaga podania wartości prawdopodobieństw *a priori* $P(C_k)$ klas. Te zwykle są nieznane, ale mogą być ewaluowane z dostępnych przykładów, a następnie modyfikowane w trakcie używania modelu. Dodatkowo podziały *nie* są ograniczone do hiperpłaszczyzn, a jednocześnie wyniki są proste do interpretacji. To wszystko powoduje, że klasyfikator Bayesa jest obecnie bardzo popularnym narzędziem predykcji (np. Bishop, 2006; Zhang, 2004; Manning *et al.*, 2008; Alpaydin, 2010);

Support Vector Machine jest z jednej strony rozwinięciem modelu perceptronu dla znalezienia optymalnej hiperpłaszczyzny rozdzielającej, z drugiej, modelem, w którym przestrzeń wejściowa jest w sposób optymalny mapowana do przestrzeni ukrytej, gdzie możliwe jest znalezienie optymalnej hiperpłaszczyzny rozdzielającej. Wykorzystywana jest tutaj funkcja jądra (ang. *kernel function*), umożliwiająca pośrednie obliczenia w przestrzeni ukrytej. Pozwala to na zdefiniowanie nieskończenie wymiarowych przestrzeni ukrytych, zwiększając prawdopodobieństwo ϕ-separowalności problemu. Wielką zaletą tego modelu jest jego bliski związek z teorią złożoności modeli nauczania (Vapnik, 1998; Scholkopf, Smola, 2002; Christiani, Shawe-Taylor, 2000).

Przewagą SVM nad warstwowymi sieciami neuronowymi jest fakt, że minimalizowana funkcja celu jest wklęsła, stąd rozwiązanie problemu optymalizacji jest możliwe. Z drugiej strony, model wymaga procesu nieliniowej optymalizacji. SVM są obecnie jednym z najczęściej wykorzystywanych modeli ze względu na możliwość aproksymacji przy jego użyciu bardzo dużej liczby problemów obliczeniowych.

A.5. Klasyfikatory złożone

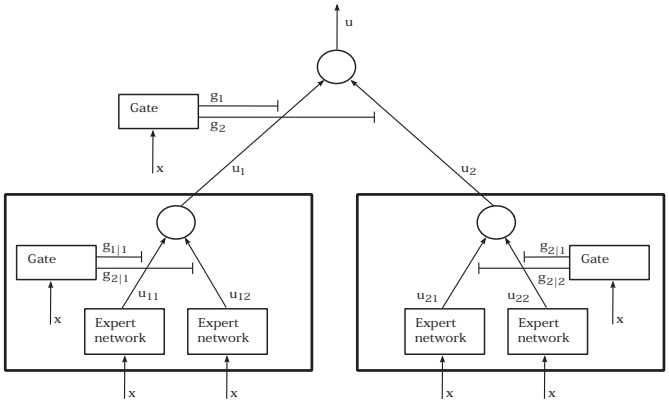
Klasyfikatory złożone starają się minimalizować całkowity błąd przede wszystkim przez minimalizację wariancji w całkowitym błędzie. Dla klasyfikacji popularnych jest co najmniej kilka modeli:

bagging jest modelem polegającym na wielokrotnym losowaniu ze zwracaniem z oryginalnego zbioru danych nowego zbioru *bootstrap* i budowaniu dla każdego takiego zbioru nowego modelu. Wynikowy model stanowi zwykłe uśrednienie dla problemu regresji czy też głosowanie dla klasyfikacji (Breiman, 1996);

komitety maszyn to ogólne określenie wielu modeli, zwykle o takiej samej architekturze, które są nauczone na podstawie różnych zbiorów uczących. Zbiory mogą być generowane niezależnie dla danego problemu, przez procedurę *bootstrap*, operacje na przestrzeni atrybutów wejściowych albo operacje na przestrzeni wyjść jak w metodzie Error Correcting Output Coding (ECOC, Dietterich, Bakiri, 1995; Valentini, Masulli, 2002). Wyniki są składane poprzez różne metody głosowania. Zbudowanie i połączenie wyników dla komitetu redukuje prawdopodobieństwo, że większość będzie się myliła,

a stąd ogranicza możliwość pomyłki. Jeśli poszczególne klasyfikatory wykorzystują lokalne metody, znalezienie wielu takich hipotez będzie zwykle prostsze niż zbudowanie jednej obejmującej cały problem (Hastie *et al.*, 2001; Duch, Itert, 2003; Bishop, 2006);

HME to system hierarchiczny – *Hierarchical Mixture of Experts* (Jordan, Jacobs, 1993; Titsias, Likas, 2002). Zadaniem HME jest taki podział przestrzeni wejściowej, żeby każda z sieci w liściach mogła dopasować prosty model dla rozwiązania tego podproblemu. Obszary podziału mogą na siebie nachodzić, stąd obiekty mogą należeć do więcej niż jednego podproblemu i być rozwiązywane przez kilka sieci eksperckich. Dzięki temu możliwe jest poprawienie skuteczności obliczania wyników.



Rysunek A.1. Architektura dwupoziomowego modelu typu HME (na podstawie (Jordan, Jacobs, 1993))

Architektura HME ma strukturę drzewiastą. W każdym liściu drzewa o indeksie i umieszczonych jest kilka sieci eksperckich (ang. *expert network*). Każda z nich otrzymuje wektor atrybutów x przykładu, generując wektor wyjściowy u_{ij}. Te wyjścia są scalane przy wykorzystaniu sieci bramkujących (ang. *gating networks*), które na wejściu otrzymują dodatkowo także wektor atrybutów x. Wyjścia liści u_i są scalane na kolejnym poziomie bliższym korzenia przez następną sieć bramkującą, dając końcowy sygnał wyjściowy u. Typowa architektura dwupoziomowego HME jest pokazana na rysunku A.1. HME ma skłonność do zbytniego dopasowywania się ze względu na zbyt dużą liczbę parametrów, co może być częściowo naprawione przez wykorzystanie apriorycznych rozkładów dla parametrów (Bishop, Svensén, 2003).

Wszystkie modele sieci eksperckich i bramkujących są z założenia *proste* (autorzy modelu HME proponują modele liniowe), są więc *słabe* w sensie poprawności niewiele lepszej niż losowa;

boosting jest metodą polegającą na budowie sekwencji klasyfikatorów tak, aby nowo utworzone poprawiały łączny błąd już zbudowanych. Najlepiej znanym algorytmem jest AdaBoost (Freund, Schapire, 1997; Friedman *et al.*, 2000). Boosting jest metodą nauczania addytywnego;

modele hierarchiczne mają zwykle na celu rozdzielenie zadania na prostsze części, rozwiązanie ich przez niezależne klasyfikatory i połączenie wyników. Często zadaniem jest także znalezienie hierarchicznie zdefiniowanych klasyfikacji (Chou, Shapiro, 2003; Ciampi *et al.*, 2007; Cesa-Bianchi *et al.*, 2006).

Wiele algorytmów, np. SVM, dopuszcza jedynie nauczanie dwuklasowe, stąd konieczny jest schemat ich rozszerzenia dla nauczania wieloklasowego. Podstawowymi są podejścia *jeden-przeciwko-wszystkim* (ang. *one-versus-all*) (Duda *et al.*, 2000), w którym dla K klas budowanych jest K, osobnych, dwuklasowych klasyfikatorów każdy rozpoznający jedną klasę; oraz *jeden-przeciwko-jednemu* (ang. *one-versus-one*) (Hastie, Tibshirani, 1998), gdzie budowane jest $K(K-1)/2$ klasyfikatorów rozróżniających każdą z par klas. Podejście *jeden-przeciwko-wszystkim* nie bierze pod uwagę podobieństw między klasami, drugie podejście wymaga budowy bardzo wielu klasyfikatorów (Kumar, Ghosh, 1999).

A.6. Agregacja wyników

Istnieje szereg różnych metod ewaluacji wyników przy wykorzystaniu komitetów (ang. *committees*, *ensembles*) klasyfikatorów. Podstawowy podział zależy od wpływu danych (Dara *et al.*, 2009; Wanas *et al.*, 2006; Oza, Tumer, 2008; Kittler *et al.*, 1997; Rokach, 2009; Hastie *et al.*, 2001) na końcowy wynik. Istnieją trzy podstawowe możliwości

- agregacja *niezależna* od danych, która wykorzystuje jedynie informacje pochodzące od poszczególnych klasyfikatorów. Tak zbudowana jest większość systemów *głosowania* i *uśredniania*: wykorzystywane są jedynie końcowe klasyfikacje. Końcowa klasyfikacja ma postać

  $$f(w_i, Cl^i(x)), \tag{A.4}$$

  gdzie w_i są ustalonymi w trakcie nauczania wagami przypisanymi klasyfikatorom. Może nimi być np. znaleziona w trakcie nauczania skuteczność. Takie systemy są często bardzo skuteczne, jednak są też podatne na błędy wynikające z faktu, że wagi klastrów pozostają niezmienne. To podejście nie uwzględnia dodatkowej informacji o rozpatrywanym wzorcu;

- agregacja *pośrednio zależna* od danych, w których wagi klasyfikatorów są *zależne* od klasyfikowanego przykładu i końcowa klasyfikacja ma postać

  $$f(w_i(Cl^j(x)), Cl^i(x)). \tag{A.5}$$

Tak więc wagi przy klasyfikacji konkretnego przykładu x są wprost od niego zależne. Waga $w_i(x)$ odpowiada, na ile klasyfikator Cl^i jest kompetentny w klasyfikacji tego przykładu (Brodowski, Podolak, 2011; Dara *et al.*, 2009). Takim typem jest właśnie opisywany w pracy model HCOC.

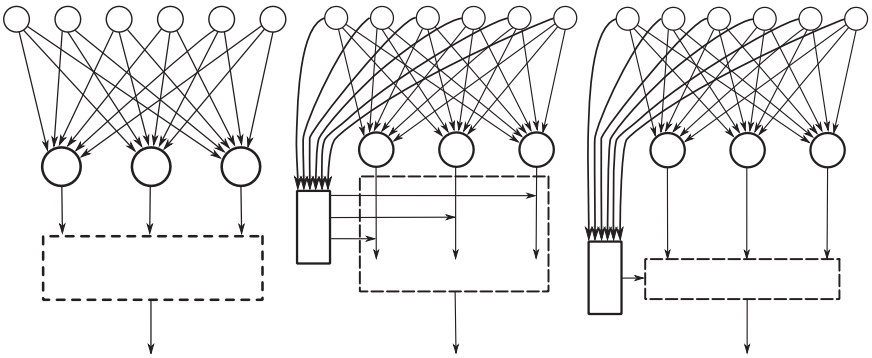
Pozwala to na skuteczniejszy wybór, jednak budowa klasyfikatora jest bardziej złożona. Dodatkowo, jeśli klasyfikatory wykazują dużą korelację w obliczaniu wyników i popełnianiu podobnych błędów, rezultaty mogą być zafałszowane. Konieczne jest więc wymuszenie różnorodności (Giacinto, Roli, 2001; Kuncheva, 2005; Tumer, Ghosh, 1996);

- agregacja *bezpośrednio* zależna od przykładów postaci

$$f(w_i(x, Cl^j(x)), Cl^i(x)), \tag{A.6}$$

uwzględniająca wprost i klasyfikacje, i atrybuty przykładów. Takimi systemami są różne metody dynamicznej selekcji klasyfikatorów opartych na wykrywaniu lokalnej skuteczności, model HME, metody oparte na cechach.

Relacje różnych typów agregacji pokazane są na rysunku A.2.



Rysunek A.2. Różne rodzaje zależności wyniku od agregacji wyników w komitetach maszyn: niezależny od danych, zależny pośrednio i zależny bezpośrednio. U góry dane i ewentualne klasyfikatory w korzeniu, pośrodku klasyfikatory pochodne, na dole moduły implementujące reguły łączenia wyników

Badania przeglądowe pokazują, że najlepiej sprawują się modele pośrednio zależne od danych: są najmniej zależne od niezgodności rozkładów zbiorów uczących i testujących, mają najwyższy stopień generalizacji. Po nich najlepsze są metody bezpośrednio zależne od danych, na końcu metody od nich niezależne (Dara *et al.*, 2009; Wanas *et al.*, 2006).

A.7. Sposoby selekcji końcowej klasy

W literaturze proponowanych jest wiele metod bezpośredniego wyboru końcowych klasyfikacji na podstawie wyników klasyfikatorów bazowych (Kittler *et al.*, 1997; Tumer, Ghosh, 1996). Kittler *et al.* (1997) rozpatrują trzy podstawowe modele agregacji wyników L klasyfikatorów Cl^l

- iloczynowy $c = \arg\max_{C_k} \Pi_{l=1}^{L} \frac{P(C_k|Cl^l)}{P(C_k|x)} P(C_k|x)$,
- sumy $c = \arg\max_{C_k} (1-L)P(C_k|x) + \sum_{l=1}^{L} P(C_k|Cl^l)$,
- mediany $c = \arg\max_{C_k} \sum_{l=1}^{L} P(C_k|Cl^l, x)$.

A.8. Podejścia przy aglomeratywnym klastrowaniu

W metodach klastrowania przez łączenie klastrów wykorzystuje się kilka podstawowych zasad łączenia klastrów:

single linkage (pojedynczego połączenia albo *najbliższego sąsiada*), gdzie odległość między klastrami $d(Q^1, Q^2)$ jest określana jako odległość między najbliższymi obiektami z dwóch klastrów

$$d(Q^1, Q^2) = \min_{\substack{i \in Q^1 \\ i' \in Q^2}} d_{ii'}.$$

Ta metoda wykazuje efekt *łańcuchowania*, łącząc szeregi obiektów parami bliskie. Tworzone klastry nie są wobec tego zwarte;

complete linkage (pełnego połączenia albo *najdalszego sąsiada*), gdzie odległość między klastrami jest definiowana jako odległość między ich najdalszymi obiektami

$$d(Q^1, Q^2) = \max_{\substack{i \in Q^1 \\ i' \in Q^2}} d_{ii'}.$$

Uzyskane klastry będą zwarte. Czasem klastry nie są wyraźnie od siebie oddzielone: niektóre obiekty z danego klastra są bliżej do obiektów z klastra sąsiedniego niż obiektów z tego samego;

group average (średnia grupowa) definiowana jako uśredniona odległość między elementami klastrów

$$d(Q^1, Q^2) = \frac{1}{\#Q^1 \#Q^2} \sum_{i \in Q^1} \sum_{i' \in Q^2} d_{ii'},$$

gdzie $\#Q^l$ oznacza liczność klastra. To pewien kompromis między poprzednimi podejściami. Wyniki klastrowania mogą jednak bardzo zależeć od numerycznych wartości reprezentacji obiektów.

A.9. Algorytm GNG

Algorytm Rosnącego Gazu Neuronowego (ang. *Growing Neural Gas*, GNG) jest przykładem nauczania samoorganizującego (Fritzke, 1995). Dla każdego wylosowanego wektora wejściowego wyszukiwany jest, wśród już utworzonych, najbliższy i drugi najbliższy „neuron" (są one aktywowane). Jeśli liczba neuronów nie przekroczyła progu, dla wylosowanego neuronu tworzony jest nowy neuron. Znalezione najbliższe neurony są przesuwane w kierunku wektora wejściowego, tworząc z nim razem klaster przez połączenie krawędziami. Krawędzie między neuronami, które nie są jednocześnie aktywowane, są usuwane (krawędzie się „starzeją"). Neurony rzadko aktywowane też są usuwane. Neurony akumulują także „błąd" obliczany jako odległość od wektora, który je aktualizuje. Jeśli dla danego neuronu błąd przekracza pewien próg, to między nim a jego najbliższym sąsiadem dodawany jest nowy neuron. Dzięki temu obszary bardziej aktywne są reprezentowane przez większą liczbę neuronów.

Dzięki takiemu postępowaniu tworzą się grupy neuronów połączonych krawędziami, które odpowiadają częstemu jednoczesnemu aktywowaniu. Tak utworzone klastry po ukończeniu nauczania mogą być analizowane i mogą być im nadane etykiety. Algorytm pokazany jest na rysunku A.3. Warunkiem końcowym pętli jest ustabilizowanie się położenia neuronów.

Zaletą algorytmu GNG jest jego zdolność do dostosowywania się do zmieniającego się środowiska wejściowego (Fritzke, 1997). Dzięki temu nadaje się bardzo dobrze do sytuacji, gdy, tak jak przy urównolegleniu nauczania i klastrowania w HCOC, wektorami wejściowymi nie są przykłady z ustalonego i skończonego zbioru, ale są wektorami ciągle zmieniających się w trakcie nauczania aktywacji $Cl(x)$.

A.10. Niektóre miary różnorodności

Miary różnorodności określają, na ile dwa (lub więcej) klasyfikatory dają różne odpowiedzi dla zbiorów danych. Gdyby wszystkie wykorzystane w złożonym modelu klasyfikatory dawały *te same* odpowiedzi, to samo składanie nie wnosiłoby żadnej nowej wartości. Znany jest cały szereg różnych miar różnorodności. Najprostsze jest podejście, gdy mamy do czynienia z dwoma klasyfikatorami Cl^1 i Cl^2. Istnieje wtedy kilka możliwości

	Cl^2 poprawnie	Cl^2 niepoprawnie
Cl^1 poprawnie	a	b
Cl^1 niepoprawnie	c	d

gdzie a, b, c, d odpowiadają odpowiednie frakcje poprawnie/niepoprawnie rozpoznanych przykładów oraz $a + b + c + d = 1$. Przy takiej definicji można zde-

Input: X – przykłady uczące
while *True* **do**
  $x \leftarrow sample(X)$ // wylosuj kolejny przykład uczący
  $n_{win} \leftarrow \arg\min_{n \in N} |x - w(n)|$ // najlepszy neuron
  $n_s \leftarrow \arg\min_{n \in N, n \neq win} |x - w(n)|$ // drugi najlepszy
  if $n_{win}, n_s \notin E$ **then**
    $E \leftarrow E \cup (n_{win}, n_s)$ // dodanie nowej krawędzie
    $age(n_{win}, n_s) \leftarrow 1$
  else
    $age(n_{win}, n_s) \leftarrow age(n_{win}, n_s) + 1$ postarzenie istniejącej krawędzi
  end
  if $age(n_{win}, n_s) > maxage$ **then**
    $E \leftarrow E \setminus (n_{win}, n_s)$ // usunięcie niewykorzystywanej krawędzi
  end
  // przysuń neuron wygrywający i jego sąsiadów do przykładu
  $n_{win} \leftarrow n_{win} + \mu_{win}(n_{win} - x)$
  for n_i:
  $(n_{win}, n_i) \in E$ **do**
    $n_i \leftarrow n_i + \mu_s(n_i - x)$
  end
  $err(n_{win}) \leftarrow err(n_{win}) + |y - w(n_{win})|$ // uaktualnienie błędu
  if *mod(iteration, R)* $= 0$ **then**
    // wykonanie co ustaloną liczbę iteracji
    $n_{err} \leftarrow \max_{n_i} err(n_i)$ // neuron z maksymalnym błędem
    if $err(n_{err}) > maxerr$ **then**
      // jeśli błąd przekracza ustalony próg, to spośród
      // sąsiadów n_{err} wyszukanie neuronu z największym błędem
      $n_{serr} \leftarrow \max_{n_i:(n_i,n_{err}) \in E} err(n_i)$
      $E \leftarrow E \setminus (n_{err}, n_{serr})$ // usunięcie krawędzi łączącej
      $n_n \leftarrow new\_node()$ // nowy neuron między n_{err} i n_{serr}
      $w(n_n) \leftarrow (w(n_{err}) + w(n_{serr}))/2$
      $age(n_{err}, n_n) \leftarrow age(n_n, n_{serr}) \leftarrow 1$
      $E \leftarrow E \cup (n_{err}, n_{new}) \cup (n_{new}, n_{serr})$ // nowe krawędzie
    end
  end
end

Rysunek A.3. Algorytm Growing Neural Gas

finiować szereg miar różnorodności dwóch lub więcej klasyfikatorów. Niektóre z nich przedstawione są w tabeli A.3.

Tabela A.3. Niektóre miary różnorodności odpowiedzi klasyfikatorów. Za: Shipp, Kuncheva (2002); Meynet, Thiran (2010); Yule (1900); Brown *et al.* (2005)

$Q_{ij} = \frac{ad - bc}{ad + bc}$	Q-statystyka
$D_{ik} = b + c$	niezgodność klasyfikatorów Cl^i i Cl^k
$\rho_{i,k} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}$	korelacja między odpowiedziami dwóch klasyfikatorów, podobne do Q
$kw = \frac{1}{NL^2} \sum_{i=1}^{N} l(x_i)(L = l(x_i))$	uśrednienie liczby zgodnych i poprawnych odpowiedzi; $l(x_i)$ jest liczbą (spośród L) klasyfikatorów poprawnie rozpoznających przykład x_i (Kohavi–Wolpert)
$\kappa = 1 - \frac{(1/L)\sum_{i=1}^{N} l(x_i)(L - l(x_i))}{N(L-1)\bar{p}(1-\bar{p})}$	miara zgodności κ, gdzie $\bar{p}$ jest średnią poprawnością klasyfikatorów; związana z miarami kw i D
$ITD = \binom{L}{2} / \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} I(Cl^i, Cl^j)$	teoretyczna różnorodność informacyjna, gdzie $I(Cl^i, Cl^j)$ jest miarą podobieństwa klasyfikatorów; odpowiada średniej różnorodności pomiędzy L klasyfikatorami

Miara różnorodności wydaje się dość oczywista i przekonująca. Brak jednak dowodów korelacji między błędem uzyskiwanym przez komitet maszyn a wyliczonymi wartościami wag. Wydaje się, że sama miara różnorodności, bez innych elementów, nie jest wystarczająca dla określenia dokładności komitetu klasyfikatorów (Kuncheva, 2009). W wielu doświadczeniach (np. Shipp, Kuncheva, 2002) dla części problemów ta korelacja jest wyraźna, dla innych już nie. Brown *et al.* (2005) zwracają uwagę na dwojaki sposób ewentualnego wykorzystania:

- pośredni, w którym algorytm tworzący model losuje przykłady uczące, aby utworzyć szereg niezależnych zbiorów i wygenerować niezależne i różnorodne klasyfikatory składowe, nie obliczając tej miary ani razu,
- bezpośredni, w którym algorytm modyfikuje zbiór uczący bezpośrednio tak, aby uzyskać różne modele; takim jest np. model boosting.

A.11. Rozkład Beta

To rozkład zmiennej losowej $\mu \in [0,1]$ opisującej prawdopodobieństwo zajścia zdarzenia binarnego: jeśli $\mu_1, \ldots, \mu_n$, są wartościami zmiennej losowej, równomiernie rozłożonymi na przedziale $[0,1]$, to rozkład $Beta(1,n)$ opisuje prawdopodobieństwo $\max(\mu_1, \ldots, \mu_n)$.

$$Beta(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{A.7}$$

$$= \frac{\mu^{a-1}(1-\mu)^{b-1}}{\int_0^1 u^{a-1}(1-\mu)^{b-1}du}, \tag{A.8}$$

gdzie

$$\Gamma(x) = \int_0^\infty u^{x-1}e^{-u}du. \tag{A.9}$$

$\Gamma(x+1) = x\Gamma(x)$, a jeśli $x \in \mathbb{N}$, to $\Gamma(x+1) = x!$. Rozkład Beta jest przypadkiem rozkładu Dirichleta dla $K = 2$.

A.12. Rozkład Dirichleta

Jest wielowymiarowym rozkładem K zmiennych losowych $0 \leq \mu_k \leq 1$, gdzie $k = 1, \ldots, K$ oraz $\sum_{k=1}^K \mu_k = 1$. Rozkład jest parametryzowany wektorem $\alpha = (\alpha_1, \ldots, \alpha_K)$. Funkcja gęstości prawdopodobieństwa ma postać

$$Dir(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)}\Pi_{k=1}^K \mu^{\alpha_k - 1}. \tag{A.10}$$

Parametry $\alpha_k > 0$, aby rozkład był zmormalizowany.

Rozkład Dirichleta jest wielowymiarowym uogólnieniem rozkładu *Beta*. Parametry α_k mogą być interpretowane jako liczby obserwacji odpowiednich zmiennych losowych μ_k. Funkcja gęstości ma skończoną wartość, pod warunkiem że $\alpha_k \geq 1$ dla każdego k.

A.13. Walidacja krzyżowa i błąd $Err^{(0.632)}$

Jeśli w trakcie budowania modelu dostępna jest nieograniczona (lub chociaż bardzo duża) liczba danych, to możliwy jest podział danych na część uczącą i część, na której wykonywane są testy. Zwykle tak nie jest i projektant musi poradzić sobie w trakcie testowania z dostępnymi danymi, dbając o to, żeby dane testujące nie były w żaden sposób używane podczas samego nauczania.

Najpopularniejszym wyborem jest tzw. walidacja krzyżowa (ang. *cross validation*), w której zbiór dostępnych danych dzielony losowo na P rozłącznych części i cykl nauczanie–testowanie powtarzany jest P razy: model tworzony jest na $P-1$ częściach i testowany na pozostałej. Na końcu wyniki testów są uśredniane. W ten sposób dane testujące nie są wykorzystywane w trakcie budowy modelu, który testują. Formalnie

$$Err^{CV} = \frac{1}{N}\sum_{i=1}^{N} \ell(t_i, \hat{f}^{(-i)}(x_i)), \tag{A.11}$$

gdzie t_i jest prawdziwą wartością dla x_i, a $\hat{f}^{(-i)}(x_i)$ oznacza ewaluację nieznanej funkcji dla x_i na zbiorze uczącym *nie* zawierającym x_i. Najczęściej wybieranymi P są 5 i 10. Większe wartości P dają mały *bias* (odchylenie średniej predykcji od średniej dla nieznanej prawdziwej funkcji F), ale za to dużą wartość wariancji (Hastie *et al.*, 2001).

Często stosowaną techniką w ewaluacji wartości statystycznych jest metoda *bootstrap*, szczególnie wtedy, gdy liczba dostępnych danych jest mała. Dla problemu, w którym dostępny jest N-elementowy zbiór przykładów X, *bootstrap* polega na losowaniu ze zwracaniem B zbiorów X_b wszystkich N-elementowych. Estymacja błędu Err^{BOOT} ma postać

$$Err^{BOOT} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} \ell(t_i, \hat{f}^{*b}(x_i)), \tag{A.12}$$

gdzie $\hat{f}^{*b}(x_i)$ oznacza ewaluację przy wykorzystaniu zbioru *bootstrap* X_b.

Prawdopodobieństwo, że dany przykład x_i pojawi się w b-tym zbiorze *bootstrap* X_b, wynosi $1-\left(1-\frac{1}{N}\right)^N \sim 1-e^{-1} = 0.632$. Błąd Err^{BOOT} jest więc zbyt optymistyczny – wykorzystuje do ewaluacji błędu przykłady, dla których budowany był model (Efron, 1983). Modyfikacja polega na opuszczeniu we wzorze tych przykładów, które były użyte do budowy aktualnego modelu

$$Err^{(1)} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{|B^{-i}|} \sum_{b\in B^{-i}} \ell(t_i, \hat{f}^{*b}(x_i)), \tag{A.13}$$

gdzie B^{-i} jest zestawem tych zbiorów *bootstrap*, które nie zawierają przykładu x_i. W ten sposób uwzględniony jest warunek, aby przykłady biorące udział w nauczaniu nie były używane do ewaluacji błędu.

Jak zauważają Efron, Tibshirani (1997), ewaluacja $Err^{(1)}$ jest pesymistyczna, stąd ich propozycja wyrównania

$$Err^{(.632)} = 0.368 \cdot \overline{err} + 0.632 \cdot Err^{(1)}, \tag{A.14}$$

gdzie $\overline{err}$ jest błędem dla zbioru trenującego. Jest on dobrze określony dla modeli, które dla zadanego zbioru uczącego dają zawsze ten sam wynik (np. modele liniowe, drzewa decyzyjne itp.). W przypadku gdy uzyskany model jest zależny także od innych parametrów, np. zbioru początkowych wag w sieciach neuronowych, potrzebne jest jego uśrednienie z kilku prób. Doświadczenia (Efron, Tibshirani, 1997) pokazują, że $Err^{(.632)}$ i jego rozszerzenie $Err^{(.632+)}$ dają najlepsze przybliżenie rzeczywistego błędu modelu.

Bibliografia

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, Cambridge, MA.

Andrews, R., Diederich, J., Tickle, A.B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, **8**(6), 373–389.

Bartlett, P.L., Jordan, M.I., McAuliffe, J.D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, **101**(473), 138–156.

Bekkerman, R., Bilenko, M., Langford, J., editors (2012). *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, Cambridge.

Bengio, Y., LeCun, Y. (2007). Scaling learning algorithms towards AI. W: L. Bottou, O. Chapelle, D. DeCoste, J. Weston, editors, *Large-Scale Kernel Machines*. MIT Press, Cambridge, MA.

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer, Berlin – Heidelberg – New York.

Bishop, C.M., Svensén, M. (2003). Bayesian hierarchical mixtures of experts. W: U. Kjaerulff, C. Meek, editors, *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, Acapulco. Morgan Kaufmann, San Mateo, CA, s. 57–64.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**, 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984). CART: Classification and Regression Trees. W: X. Wu, V. Kumar, editors, *The top ten algorithms in data mining*. Chapman & Hall/CRC Press, New York, s. 179–201.

Brodatz, P. (1966). *Textures: A Photographic Album for Artists and Designers*. Dover Publications, New York.

Brodowski, S., Podolak, I.T. (2011). Hierarchical estimator. *Expert Systems with Applications*, **38**(10), 12237–12248.

Brown, G., Wyatt, J., Harris, R., Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, **6**, 5–20.

Brugger, D., Bogdan, M., Rosenstiel, W. (2008). Automatic cluster detection in Kohonen's SOM. *IEEE Transactions on Neural Networks*, **19**(3), 442–459.

Bühlmann, P., Yu, B. (2006). Sparse boosting. *Journal of Machine Learning Research*, **7**, 1001–1024.

Casasent, D., Wang, Y.-C.F. (2005). A hierararchical classifier using new support vector machines for automatic target recognition. *Neural Networks*, **18**, 541–548.

Ĉerný, J., Pirická, A., Rosenauerová, B. (1971). On directable automata. *Kybernetika*, **7**(4), 289–298.

Cesa-Bianchi, N., Gentile, C., Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, **7**, 31–54.

Chou, Y.-Y., Shapiro, L.G. (2003). A hierarchical multiple classifier learning algorithm. *Pattern Analysis & Applications*, **6**(2), 150–168.

Christiani, N., Shawe-Taylor, J. (2000). *Support Vector Machines and other Kernel Based Learning Methods*. Cambridge University Press, Cambridge.

Ciampi, A., Lechevallier, Y., Limas, M.C., Marcos, A.G. (2007). Hierarchical clustering of subpopulations with a dissimilarity based on the likelihood ratio statistic: application to clustering massive data sets. *Pattern Analysis and Applications*, **11**(2), 199–220.

Dara, R.A., Kamel, M.S., Wanas, N. (2009). Data dependency in multiple classifier systems. *Pattern Recognition*, **42**(7), 1260–1273.

Day, W., Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, **1**(1), 7–24.

Dean, J., Ghemawat, S. (2008). MapReduce : simplified data processing on large clusters. *Communications of the ACM*, **51**(1), 1–13.

Demŝar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**, 1–30.

Dietterich, T.G., Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, **2**, 263–286.

Duch, W., Itert, Ł. (2003). Committees of undemocratic competent models. W: *International Conference on Artificial Neural Networks*, Istambuł. Springer, Berlin – Heidelberg – New York, s. 33–36.

Duch, W., Wieczorek, T., Biesiada, J., Blachnik, M. (2004a). Comparison of feature ranking methods based on information entropy. W: *Proceedings of the IEEE International Joint Conference on Neural Networks*, Budapeszt. IEEE, New York, s. 1415–1419.

Duch, W., Setiono, R., Żurada, J.M. (2004b). Computational intelligence methods for rule-based data understanding. *Proceedings of the IEEE*, **92**(5), 771–805.

Duda, R.O., Hart, P.E., Stork, D.G. (2000). *Pattern Classification*. Wiley Interscience, Hoboken, NJ.

Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross–validation. *Journal of the American Statistical Association*, **78**(382), 316–331.

Efron, B., Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, **92**(438), 548–560.

Eibl, G., Pfeiffer, K.-P. (2005). Multiclass boosting for weak classifiers. *Journal of Machine Learning*, **6**, 189–210.

Ferguson, T.S. (1973). A bayesian analysis of some nonparametric problems. *Annals of Statistics*, **1**(2), 209–230.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**(2), 179–188.

Frank, E., Hall, M., Pfahringer, B. (2003). Locally weighted Naive Bayes. W: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Acapulco. Morgan Kaufmann, San Mate, CA, s. 249–256.

Freund, Y., Schapire, R.E. (1997). A decision theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55**, 119–139.

Friedman, J., Hastie, T., Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28**(2), 337–407.

Friedman, J., Hastie, T., Rosset, S., Tibshirani, R., Zhu, J. (2004). Discussion of three boosting papers. *Annals of Statistics*, **32**(1), 102–107.

Friedmann-Hill, E. (2003). Jess, the rule engine for Java platform. Technical report, Sandia National Laboratory, Livermore, CA.

Fritzke, B. (1993). Kohonen Feature Maps and Growing Cell Structures: a performance comparison. W: S.J. Hanson, J.D. Cowan, C.L. Giles, editors, *Advances in Neural Information Processing Systems*, vol. 5. Morgan Kaufmann, San Mateo, CA, s. 115–122.

Fritzke, B. (1995). A Growing Neural Gas network learns topologies. W: G. Tesauro, D.S. Touretzky, T.K. Leen, editors, *Advances in Neural Information Processing Systems*, vol. 7. MIT Press, Cambridge, MA, s. 625–632.

Fritzke, B. (1997). A self-organizing network that can follow non-stationary distributions. W: *International Conference on Artificial Neural Networks*, Lousanne. Springer, Berlin – Heidelberg – New York, s. 613–618.

Giacinto, G., Roli, F. (2001). An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, **22**, 25–33.

Giarratano, J., Riley, G. (1994). *Expert Systems: Principles and Programming*. PWS Publishing, Boston, MA.

Hand, D., Mannila, H., Smyth, P. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA.

Hastie, T., Tibshirani, R. (1998). Classification of pairwise couplings. W: *Advances in Neural Information Processing Systems*, vol. 10. MIT Press, Cambridge, MA, s. 507–513.

Hastie, T., Tibshirani, R., Friedman, J. (2001). *The Elements of Statistical Learning*. Springer, Berlin – Heidelberg – New York.

Haykin, S. (2009). *Neural Networks: a Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ.

Hinton, G.E., Osindero, S. (2006). A fast learning algorithm for Deep Belief Networks. *Neural Computation*, **18**, 1527–1554.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA.

Ibrahim, S., Jin, H., Lu, L., Qi, L., Wu, S., Shi, X. (2009). Evaluating MapReduce on virtual machines: the Hadoop case. W: M.G. Jaatun, G. Zhao, C. Rong, editors, *Cloud Computing*, vol. 5931 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 519–528.

Itti, L., Baldi, P. (2005). Bayesian surprise attracts human attention. W: Y. Weiss, B. Schölkopf, J. Platt, editors, *Advances in Neural Information Processing Systems*, vol. 18. MIT Press, Cambridge, MA, s. 547–554.

Jordan, M.I., Jacobs, R.A. (1993). Hierarchical Mixtures of Experts and the EM algorithm. A.I. Memo 1440, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Kari, J. (2003). Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, **295**(1–3), 432–438.

Kearns, M.J., Valiant, L.G. (1989). Cryptographic limitations on learning boolean formulae and finite automata. W: *Proceedings of 21st Annual ACM Symp. on Theory of Computing*, Seattle, WA. ACM, New York, s. 434–444.

Kearns, M.J., Valiant, L.G. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, **41**(1), 67–95.

Kittler, J., Hojjatoleslami, A., Windeatt, T. (1997). Strategies for combining classifiers employing shared and distinct pattern representations. *Pattern Recognition Letters*, **18**, 1373–1377.

Kohavi, R., Quinlan, J.R. (2002). *Decision–tree discovery*. Oxford University Press, New York, NY, s. 267–276.

Kohonen, T., Honkela, T. (2007). Kohonen network. *Scholarpedia*, **2**(1), 1568.

Kulkarni, A.V. (1978). On the mean accuracy of hierarchical classifiers. *IEEE Transactions on Computers*, **27**, 771–776.

Kumar, S., Ghosh, J. (1999). GAMLS: A Generalized framework for Associative Modular Learning Systems. W: *Proceedings of the Applications and Science of Computational Intelligence II*, vol. 3722, Orlando, FL. SPIE, Bellingham, WA, s. 24–34.

Kumar, S., Ghosh, J., Crawford, M.M. (1999). A versatile framework for labeling imagery with a large number of classes. W: *International Joint Conference on Neural Networks*, Washington, DC. IEEE, New York, NY, s. 2829–2833.

Kumar, S., Ghosh, J., Crawford, M.M. (2002). Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis & Applications*, **5**(2), 210–220.

Kuncheva, L.I. (2000). Clustering–and–selection model for classifier combination. W: *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, Brighton, UK. IEEE, New York, s. 185–188.

Kuncheva, L.I. (2005). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley Interscience, Hoboken, NJ.

Kuncheva, L.I. (2009). That elusive diversity in classifier ensembles. *Pattern Recognition and Image Analysis*, **2652**, 1126–1138.

Kurzyński, M. (1983). Decision rules for a hierarchical classifier. *Pattern Recognition Letters*, **1**, 305–310.

Manning, C.D., Raghavan, P., Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

Meynet, J., Thiran, J.-P. (2010). Information theoretic combination of pattern classifiers. *Pattern Recognition*, **43**(10), 3412–3421.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin – Heidelberg – New York.

Minsky, M., Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA.

Nene, S.A., Nayar, S.K., Murase, H. (1996). Columbia Object Image Library (COIL–100). Technical Report CUCS–006–96, Columbia University, New York, NY.

Newman, D., Hettich, S., Blake, C., Aha, D., Murphy, P. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html.

Nguyen, A.V., Hudson, D.L., Cohen, M.E. (2010). Hadoop and MapReduce for the storage and processing of physiological data. W: *International Conference on Software Engineering and Data Engineering*, San Francisco, CA. ISCA, s. 366–369.

Olschewski, J., Ummels, M. (2010). The complexity of finding reset words in finite automata. *Computer Research Repository*, **abs/1004.3246**.

Oza, N.C., Tumer, K. (2008). Classifier ensembles: select real world applications. *Information Fusion*, **9**, 4–20.

Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J. (2012). MapReduce and its application to massively parallel learning of decision tree ensembles. W: R. Bekkerman, M. Bilenko, J. Langford, editors, *Scaling up machine learning: parallel and distributed approaches*. Cambridge University Press, Cambridge, s. 23–48.

Parker, J.R. (2001). Rank and response combination from confusion matrix data. *Information Fusion*, **2**(2), 113–120.

Podolak, I.T. (2007). Hierarchical rules for a hierarchical classifier. W: *Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms*, vol. 4431 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 749–757.

Podolak, I.T. (2008). Hierarchical classifier with overlapping class groups. *Expert Systems with Applications*, **34**(1), 673–682.

Podolak, I.T., Bartocha, K. (2009). A hierarchical classifier with Growing Neural Gas clustering. W: *Adaptive and Natural Computing Algorithms*, vol. 5495 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 283–292.

Podolak, I.T., Roman, A. (2009). A new notion of weakness in classification theory. W: *Advances in Intelligent and Soft Computing*, no. 57 in AISC. Springer, Berlin – Heidelberg – New York, s. 239–245.

Podolak, I.T., Roman, A. (2011a). Fusion of supervised and unsupervised training methods for a multi-class classification problem. *Pattern Analysis and Applications*, **14**(4), 395–413.

Podolak, I.T., Roman, A. (2011b). Risk estimation for hierarchical classifier. W: *Hybrid Artificial Intelligent Systems*, no. 6678 in LNCS. Springer, Berlin – Heidelberg – New York, s. 156–163.

Podolak, I.T., Roman, A. (2011c). Risk function estimation for subproblems in a hierarchical classifier. *Pattern Recognition Letters*, **32**, 2136–2142.

Podolak, I.T., Roman, A. (2012). Theoretical foundations and experimental results for a hierarchical classifier with overlapping clusters. *Computational Intelligence*. http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8642.2012.00469.x/abstract, w druku.

Podolak, I.T., Xi, D., Lee, S.-W. (2002). Facial component extraction and face recognition with support vector machines. W: *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, Barcelona. IEEE, IEEE Computer Society, New York, s. 76–81.

Podolak, I.T., Biel, S., Bobrowski, M. (2006). Hierarchical classifier. W: R. Wyrzykowski, editor, *Parallel Processing and Applied Mathematics*, vol. 3911 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 591–598.

Podolak, I.T., Roman, A., Jędrzejczyk, D. (2012a). Application of hierarchical classifier to minimal synchronizing word problem. W: L. Rutkowski, editor, *Lecture Notes in Computer Science*, vol. 7267 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 421–429.

Podolak, I.T., Roman, A., Deszyńska, A. (2012b). On the number of clusterings in a hierarchical classification model with overlapping clusters. *Schedae Informaticae*, **20**, 137–158.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Quinlan, J.R. (1996). Learning decision tree classifiers. *ACM Computing Surveys*, **28**, 71–72.

R Development Core Team (2005). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna.

Rokach, L. (2009). Taxonomy for characterizing ensemble methods in classification tasks: a review and annotated bibliography. *Computational Statistics and Data Analysis*, **53**, 4046–4072.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536.

Samaria, F., Harter, A. (1994). Parameterisation of a stochastic model for human face identification. W: *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, Sarasota, FL. IEEE, New York, s. 138–142.

Schaefer, R., Telega, H. (2007). *Foundations of Global Genetic Optimization*, vol. 74 of *Studies in Computational Intelligence*. Springer, Berlin – Heidelberg – New York.

Schapire, R.E. (1990). The strength of weak learnability. *Machine Learning*, **5**, 197–227.

Schapire, R.E., Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3), 297–336.

Scholkopf, B., Smola, A.J. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA.

Schraudolph, N.N., Yu, J., Gunter, S. (2007). A stochastic quasi–newton method for on–line convex optimization. W: M. Meila, X. Shen, editors, *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico. Journal of Machine Learning Research, s. 436–443.

Schuerman, J., Doster, W. (1984). A decision theoretic approach to hierarchical classifier design. *Pattern Recognition*, **17**(3), 359–369.

Sethi, I.K., Sarvarayudu, G. P.R. (1982). Hierarchical classifier design using mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **4**(4), 441–445.

Setiono, R. (2001). Feedforward neural network construction using cross validation. *Neural Computation*, **13**, 2865–2877.

Setiono, R., Leow, W.K. (2000). FERNN: an algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, **12**(1/2), 15–25.

Shipp, C.A., Kuncheva, L.I. (2002). Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, **3**(2), 135–148.

Sneath, P.H., Sokal, R.R. (1973). *Principles of Numerical Taxonomy*. Freeman, San Francisco, CA.

Tapia, E., Bulacio, P., Angelone, L. (2010). Recursive ECOC classification. *Pattern Recognition Letters*, **31**(3), 210–215.

Ting, K.M., Wells, J.R., Tan, S.C., Teng, S.W., Webb, G.I. (2010). Feature-subspace aggregating: ensembles for stable and unstable learners. *Machine Learning*, **82**(3), 375–397.

Titsias, M.K., Likas, A. (2002). Mixture of Experts classification using a hierarchical mixture model. *Neural Computation*, **2244**, 2221–2244.

Tou, J.Y., Tay, Y.H., Lau, P.Y. (2007). Gabor filters and grey-level co-occurence matrices in texture classification. W: *Proceedings of the MMU International Symposium on Information and Communications Technologies*.

Trahtman, A.N. (2006). An efficient algorithm finds noticeable trends and examples concerning the Ĉerný conjecture. W: *Lecture Notes in Computer Science*, vol. 4162 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 789–800.

Tsapanos, N., Tefas, A., Nikolaidis, N., Pitas, I. (2011). Shape matching using a binary search tree structure of weak classifiers. *Pattern Recognition*, **45**(6), 2363–2376.

Tumer, K., Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science*, **8**(3–4), 385–404.

Valentini, G., Masulli, F. (2002). Ensembles of learning machines. W: R. Tagliaferri, M. Marinaro, editors, *Neural Nets, WIRN*, vol. 2486 of *LNCS*. Springer, Berlin – Heidelberg – New York, s. 3–19.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM*, s. 1134–1142.

Vapnik, V.N. (1996). *The Nature of Statistical Learning Theory*. Springer, Berlin – Heidelberg – New York.

Vapnik, V.N. (1998). *Statistical Learning Theory*. Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control. Wiley-Interscience, Hoboken, NJ.

Wanas, N., Dara, R.A., Kamel, M.S. (2006). Adaptive fusion and co-operative training for classifier ensembles. *Pattern Recognition*, **39**(9), 1781–1794.

Wang, Y.-C.F., Casasent, D. (2008). New support vector-based design method for binary hierarchical classifiers for multi-class classification problems. *Neural Networks*, **21**(2–3), 502–510.

Wang, Y.-C.F., Casasent, D. (2009). A support vector hierarchical method for multiclass classification and rejection. W: *Proceedings of International Joint Conference on Neural Networks*, Atlanta, GA. IEEE, New York, s. 3281–3288.

Witten, I.H., Frank, E. (2000). *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Mateo, CA.

Wołoszynski, T., Kurzyński, M. (2011). A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, **44**(10–11), 2656–2668.

Yule, G.U. (1900). On the association of attributes in statistics: with illustrations from the material of the childhood society, &c. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **194**(252–261), 257–319.

Zhang, H. (2004). The optimality of Naive Bayes. W: V. Barr, M. Zdravko, editors, *Proceedings of the Seventeenth International Florida Intelligence Research Society*, Miami Beach, FL. AAAI Press.

Zwitter, M., Sokolic, M. (1988). Primary tumor data set. dane przekazane przez University Medical Center, Institute of Oncology, Ljubljana.

Skorowidz pojęć

REDAKTOR PROWADZĄCY
Jadwiga Makowiec

ADIUSTACJA JĘZYKOWO-STYLISTYCZNA I KOREKTA
Jerzy Hrycyk

Wydawnictwo Uniwersytetu Jagiellońskiego
Redakcja: ul. Michałowskiego 9/2, 31-126 Kraków
tel. 12-631-18-80, tel./fax 12-631-18-83