Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

PRACA DYPLOMOWA MAGISTERSKA
MATEMATYKA

DRZEWA KLASYFIKACYJNE ICH BUDOWA, PROBLEMY ZŁOŻONOŚCI I SKALOWALNOŚCI

Autor:
Mariusz Gromada

Promotor:
Prof. dr hab. Jacek Koronacki

WARSZAWA, STYCZEŃ 2006

. . . . . .

Podpis promotora

. . . . . .

Podpis dyplomanta

Spis treści

Wstęp

Temat pracy dotyczy problemu dyskryminacji oraz budowy drzew klasyfikacyjnych w kontekście ich przydatności do rozwiązywania zadań o dużym wymiarze prób losowych i/lub dużym wymiarze wektora obserwacji, w których podstawowego znaczenia nabiera złożoność obliczeniowa drzewa. Radzenie sobie z dużymi zbiorami danych wymaga konstrukcji specjalnych technik sortowania danych w trakcie budowy drzewa, kodowania, organizacji wzrostu i przycinania drzewa. Wymaga także zrównoleglenia obliczeń. Przedmiotem pracy jest sformułowanie modelu analizy dyskryminacyjnej oraz analiza możliwych rozwiązań podanych zagadnień, wraz z implementacją jednego z nich. Autorskim wkładem do pracy są rozdziały 1, 2 oraz 5.

W pierwszym rozdziale omawiam problem dyskryminacji pod nadzorem, nazywanej analizą dyskryminacyjną. Autorskim wkładem jest wprowadzenie formalnego modelu klasyfikacyjnego osadzonego w przestrzeni probabilistycznej wraz z twierdzeniami o numerach: 1.3.1, 1.3.2, 1.4.1.

Rozdział drugi poświęcony jest budowie drzew klasyfikacyjnych. Szczególną uwagę zwracam na problem złożoności i skalowalności. Autorskim wkładem jest wprowadzenie formalnej definicji drzewa klasyfikacyjnego w oparciu o podstawy teorii grafów oraz o model klasyfikacyjny przedstawiony w rozdziale pierwszym. Podaję oraz dowodzę twierdzenia: 2.4.1 i 2.3.1. Dodatkowo omawiam nowatorską technikę przycinania drzew wykorzystującą zasadę minimalnej długości kodu, MDL (M. Mehta, J. Rissanen, R. Agrawal, 1995).

W rozdziale trzecim i czwartym skupiam się na przedstawieniu indukcji drzew decyzyjnych metodą SLLIQ (M. Mehta, R. Agrawal, J. Rissanen, 1996) oraz SPRINT (J.C. Shafer, R. Agrawal, M. Mehta, 1996).

Rozdział piąty prezentuje implementację klasyfikatora SLIQ wraz z implementacją przycinania drzew metodą MDL. Implementację przeprowadziłem we współpracy z Instytutem Podstaw Informatyki Polskiej Akademii Nauk w ramach rozwoju pakietu „`dmLab`". Tekst rozdziału zawiera również analizę złożoności czasowej i skalowalności implementacji.

Pracę kończą dodatki A i B, w których zebrałem podstawowe pojęcia wykorzystane w tekście z topologii, teorii miary, probabilistyki oraz teorii grafów.

Rozdział 1

Analiza dyskryminacyjna

1.1. Wprowadzenie

Postęp informatyzacji życia codziennego umożliwił przechowywanie i przetwarzanie olbrzymich ilości danych[1] oraz odkrywanie ukrytych w nich zależności[2]. Nowoczesnych metod analizy danych dostarcza współczesna statystyka matematyczna. Szczególnie praktycznie znaczenie mają metody klasyfikacyjne. Klasyfikacja jest dziś bardzo szeroko wykorzystywana przez świat nauki, biznesu, przemysłu oraz medycyny. Klasyfikacja (w skrócie) polega na poszukiwaniu najlepszego możliwego rozdzielenia obserwacji z różnych populacji. Wyróżnia się dwie metody klasyfikacji. Pierwsza z nich to klasyfikacja bez nadzoru, nazywana analizą skupień. Druga, wykorzystująca próby uczące, określana jest mianem klasyfikacji pod nadzorem lub też analizą dyskryminacyjną. W poniższej pracy ograniczamy się do przypadku drugiego.

Próbę uczącą stanowić może każdy zbiór obserwacji ze znanym podziałem na populacje (klasy). Dyskryminacja poszukuje najlepszej reguły klasyfikacyjnej, która każdej nowej obserwacji (z nieznaną przynależnością do populacji) przypisze pewną klasę. W dalszej części pracy konstruujemy precyzyjny model klasyfikacyjny, wyróżniając w nim rodzinę reguł najefektywniejszych.

1.2. Problem skal pomiarowych

W teorii pomiaru rozróżnia się 4 podstawowe *skale pomiaru*, wprowadzone przez Stevensa (1959), uporządkowane od najsłabszej do najmocniejszej: *nominalna*, *porządkowa* (*rangowa*), *przedziałowa* (*interwałowa*), *ilorazowa* (*stosunkowa*). Podstawowe własności skal pomiaru przedstawia tabela 1.1.

Przyrządy, dokonujące pomiarów w praktyce, posiadają skończoną dokładność. Wykorzystując to spostrzeżenie, konstrukcję modelu klasyfikacyjnego ograniczyliśmy do rozpatrywania przeliczalnych przestrzeni możliwych wartości pomiarów.

[1] Przechowywanie, dostarczenie i przetwarzanie danych opisuje tematyka hurtowni danych (ang. Data Warehousing).

[2] Odkrywaniem zależności w zbiorach danych zajmuje się tematyka eksploracji danych (ang. Data Mining).

Typ skali	Dopuszczalne relacje	Dopuszczalne operacje arytmetyczne
Nominalna	równości, różności	zliczanie zdarzeń (liczba relacji równości, różności)
Porządkowa	powyższe oraz większości i mniejszości	zliczanie zdarzeń (liczba relacji równości, różności, większości, mniejszości)
Przedziałowa	powyższe oraz równości różnic i przedziałów	powyższe oraz dodawanie i odejmowanie
Ilorazowa	powyższe oraz równości ilorazów	powyższe oraz mnożenie i dzielenie

Tabela 1.1: Podstawowe własności skal pomiarowych

1.3. Model klasyfikacyjny

Model, jako ogólne (teoretyczne) odzwierciedlenie różnych aspektów rzeczywistości, jest strukturą opartą o pewien system założeń (aksjomatów), pojęć i relacji między nimi. Mianem własności modelu określamy każdą logiczną (najczęściej sprawdzalną empirycznie) konsekwencję sformułowanych wcześniej aksjomatów. Poniżej przedstawiamy konstrukcję modelu klasyfikacyjnego, który umieszcza klasyfikację z próbą uczącą w szeroko rozumianej probabilistyce. Wykorzystując przestrzenie topologiczne, mierzalne i probabilistyczne, wprowadzamy szereg nowych definicji, zwracając przy tym uwagę na kwestie szczególnie istotne. Często odnosimy się do dodatku „A", który w sposób podstawowy omawia materiał zapożyczony z innych działów matematyki.

1.3.1. Zbiór przykładów i pojęcie atrybutu

Niech $\{X_k\}$, $k = 1, \ldots, p$ będzie dowolną rodziną niepustych zbiorów.

Definicja 1.3.1 *Zbiór:*

$$X := X_1 \times X_2 \times \ldots \times X_p \tag{1.1}$$

nazywamy zbiorem przykładów. Każdy element zbioru X nazywamy krótko przykładem.

Z każdym zbiorem X_k wiążemy funkcję:

$$A_k : X \to X_k \quad \text{gdzie} \quad X \ni x = (x_1, x_2, \ldots, x_p) \mapsto x_k \in X_k \tag{1.2}$$

Innymi słowy $\forall\ x \in X,\ \forall\ k \in \{1, \ldots, p\}$ mamy $A_k(x) = x_k$, oraz $A_k(X) = X_k$

Definicja 1.3.2 *Przekształcenie A_k nazywamy $k-tym$ atrybutem w X. Zbiór $A_k(X) = X_k$ nazywamy zbiorem wartości atrybutu A_k.*

Definicja 1.3.3 *Jeżeli zbiór X_k jest skończony i nieuporządkowany, to atrybut A_k nazywamy nominalnym. W przypadku przeliczalnego i uporządkowanego zbioru X_k mówimy o porządkowym atrybucie A_k.*

Atrybut nominalny może być atrybutem porządkowym, podobnie pewnego rodzaju atrybuty porządkowe można traktować jako atrybuty nominalne. Zbliżone pojęcie atrybutu wprowadzone jest w [2].

Przykładem atrybutu nominalnego może być kolor (zielony, żółty,...). Atrybutem porządkowym jest np. wzrost podany w centymetrach.

1.3.2. Rodzina obserwacji

Zbiór X określa wszystkie możliwe wartości ustalonego zestawu atrybutów. Każdy element $x = (x_1, x_2, \ldots, x_p) \in X$, gdzie $x_k \in X_k$ jest wartością atrybutu A_k, nazwaliśmy przykładem. Przykład jest elementem unikalnym w X. Dalej wprowadzimy pewną rodzinę opartą na X, która pozwoli myśleć o przykładach w kategoriach rozkładów.

Definicja 1.3.4 *Rodziną obserwacji nazywamy każdą niepustą rodzinę przykładów*

$$\mathcal{X}^{\mathcal{I}} := \{x^i\}_{i \in \mathcal{I}} \tag{1.3}$$

gdzie $\mathcal{I}$ jest zbiorem indeksów oraz $x^i \in X$ dla każdego $i \in \mathcal{I}$.

$\mathcal{X}^{\mathcal{I}}$ jest rodziną elementów ze zbioru X indeksowaną zbiorem indeksów $\mathcal{I}$. Pozwala to zamiast $\{x^i\} \in \mathcal{X}^{\mathcal{I}}$ pisać krótko $i \in \mathcal{I}$ (myślimy o indeksie, a nie o konkretnym przykładzie z rodziny). Ponadto dla różnych $i_1, i_2 \in \mathcal{I}$ może zachodzić $x^{i_1} = x^{i_2}$. Przykład należący do rodziny obserwacji nie musi być w niej unikalny (przypomnijmy, że zbiór przykładów taką własność posiada).

Definicja 1.3.5 *Zbiór*

$$\mathcal{X} := \bigcup_{i \in \mathcal{I}} \{x^i\} \tag{1.4}$$

nazywamy zbiorem przykładów rodziny obserwacji $\mathcal{X}^{\mathcal{I}}$.

Wniosek 1.3.1 $\mathcal{X} \subseteq X$.

Przykład to inaczej pewien zestaw wartości atrybutów. Atrybuty w rzeczywistości służą do opisu obiektów. Różne obiekty, opisywane tym samym zestawem atrybutów, mogą być identyczne w sensie wartości atrybutów, ale to nadal różne obiekty (takie obiekty opisuje ten sam przykład). Wyobraźmy sobie dwie osoby, w wieku 30 lat, z wykształceniem wyższym, które pochodzą z Warszawy. Jeżeli przyjmiemy, że do opisu osób używamy zestawu atrybutów (*wiek,wykształcenie,pochodzenie*), to jeden przykład (30,*wyższe,Warszawa*) będzie odpowiadał dwóm różnym osobom. Rodzina obserwacji umożliwia odzwierciedlenie takiej sytuacji (w przeciwieństwie do zbioru przykładów).

Definicja 1.3.6 *Obiektem (indeksem obserwacji) nazywamy każdy indeks $i \in \mathcal{I}$ należący do zbioru indeksów rodziny obserwacji $\mathcal{X}^{\mathcal{I}}$. Zbiór $\mathcal{I}$ nazywamy zbiorem obiektów.*

Podkreślmy, że każdy obiekt $i \in \mathcal{I}$ jest opisany przykładem $x^i \in X$.

1.3.3. Warstwa przykładu i przestrzeń obiektów

Definicja 1.3.7 *Warstwą przykładu $x \in \mathcal{X}$ w zbiorze obiektów $\mathcal{I}$ nazywamy zbiór:*

$$\mathcal{I}_x := \{i \in \mathcal{I} \ : \ x^i = x\} \tag{1.5}$$

Warstwa przykładu $x \in \mathcal{X}$ jest podzbiorem zbioru obiektów $\mathcal{I}$ opisanych tym samym przykładem x.

Wniosek 1.3.2 *Rodzina wszystkich warstw $\mathcal{I}^{\mathcal{X}} := \{\mathcal{I}_x\}_{x\in\mathcal{X}}$ wyznacza podział $\mathcal{I}$ na niepuste parami rozłączne podzbiory.*

Rozważmy przestrzeń topologiczną $(\mathcal{I}, \mathfrak{T})$, gdzie $\mathfrak{T}$ dowolna topologia (def. A.1.1) w $\mathcal{I}$ zawierająca rodzinę $\mathcal{I}^{\mathcal{X}}$ ($\mathcal{I}^{\mathcal{X}} \subseteq \mathfrak{T}$). Niech dalej $\mathfrak{B}$ oznacza σ-ciało borelowskie (def. A.2.2) w przestrzeni $(\mathcal{I}, \mathfrak{T})$.

Definicja 1.3.8 *Jeżeli P jest taką miarą probabilistyczną na $(\mathcal{I}, \mathfrak{B})$, że:*

$$\forall\ x \in \mathcal{X} \quad P(\mathcal{I}_x) > 0$$

to przestrzeń probabilistyczną

$$\mathfrak{I} := (\mathcal{I}, \mathfrak{B}, P)$$

nazywamy przestrzenią obiektów.

Warstwa każdego przykładu jest zbiorem mierzalnym z niezerowym prawdopodobieństwem. Fakt ten oznacza niezerowe prawdopodobieństwo wystąpienia każdego zestawu atrybutów ze zbioru $\mathcal{X}$ w przestrzeni obiektów (inaczej w $\mathcal{I}$ każdego przykładu $x \in \mathcal{X}$). Powyższe założenia pozwalają wyprowadzić pewne bardzo ważne twierdzenie.

Twierdzenie 1.3.1 *Rodzina warstw $\mathcal{I}^{\mathcal{X}}$ w przestrzeni obiektów $\mathfrak{I}$ jest co najwyżej przeliczalna. Formalnie:*

$$|\mathcal{I}^{\mathcal{X}}| \leqslant \aleph_0$$

Dowód: Twierdzenie udowodnimy nie wprost zakładając, że istnieje przestrzeń obiektów $\mathfrak{I}$, w której zachodzi $|\mathcal{I}^{\mathcal{X}}| > \aleph_0$.

Rodzina $\mathcal{I}^{\mathcal{X}}$ określa podział zbioru obiektów $\mathcal{I}$ na niepuste parami rozłączne podzbiory. Z definicji przestrzeni obiektów wiemy, że:

$$\forall\ x \in \mathcal{X} \quad P(\mathcal{I}_x) > 0$$

Niech

$$\mathcal{X}_n := \left\{x \in \mathcal{X} \mid P(\mathcal{I}_x) > \frac{1}{n}\right\}$$

Oczywiście

$$\bigcup_{n=1}^{\infty} \mathcal{X}_n = \mathcal{X}$$

Zbór $\mathcal{X}$ jest równoliczny z rodziną $\mathcal{I}^{\mathcal{X}}$, z założenia jest więc nieprzeliczalny. Musi zatem istnieć $k \in \mathbb{N}$, że $\mathcal{X}_k$ jest nieskończony: $|\mathcal{X}_k| \geqslant \aleph_0$. W szczególności można wybrać ciąg różnych przykładów: $x^1, x^2, \ldots \in \mathcal{X}_k$. Z definicji zbioru $\mathcal{X}_k$ wynika, że:

$$\forall\ n \in \mathbb{N} \quad P(\mathcal{I}_{x^n}) > \frac{1}{k}$$

Zauważmy, że rodzina $\{\mathcal{I}_{x^n}\}_{n\in\mathbb{N}}$ jest przeliczalną rodziną zbiorów parami rozłącznych. Zatem:

$$1 = P(\mathcal{I}) \geqslant \sum_{n=1}^{\infty} P(\mathcal{I}_{x^n}) > \sum_{n=1}^{\infty} \frac{1}{k} = \infty$$

Założenie okazało się nieprawdziwe, co kończy dowód.

■

Twierdzenie 1.3.1 daje pewien obraz struktury przestrzeni obiektów. Wykazaliśmy, że zbiór przykładów rodziny $\mathcal{X}^{\mathcal{I}}$, która wchodzi w skład przestrzeni obiektów $\mathfrak{I}$, może być co najwyżej przeliczalny. Oczywiście sama rodzina $\mathcal{X}^{\mathcal{I}}$ (a zatem i zbiór obiektów $\mathcal{I}$) może być nieprzeliczalna.

Definicja 1.3.9 *Warstwą atrybutu A_k, $k \in \{1,\ldots,p\}$ dla wartości $w \in X_k$ nazywamy zbiór*

$$A_k^w := \Big\{ i \in \mathcal{I} \ : \ A_k(x^i) = w \Big\} \tag{1.6}$$

Warstwa A_k^w reprezentuje obiekty, dla których wartość atrybutu A_k wynosi $w \in X_k$. Podkreślmy, że warstwa atrybutu może być zbiorem pustym.

Zauważmy, że

$$A_k^w = \bigcup_{x\in\mathcal{X},\ A_k(x)=w} \mathcal{I}_x \tag{1.7}$$

Wniosek 1.3.3 *Korzystając z aksjomatu (T_3) topologii o „otwartości unii dowolnej rodziny zbiorów otwartych" (def. A.1.1) stwierdzamy, że warstwa A_k^w, $k \in \{1,\ldots,p\}$ atrybutu A_k dla wartości $w \in X_k$ jest zbiorem mierzalnym ($A_k^w \in \mathfrak{B}$) w przestrzeni obiektów $\mathfrak{I}$.*

Dzięki powyższemu możliwe jest określenie prawdopodobieństwa przyjęcia wartości $w \in X_k$ przez atrybut A_k:

$$P(A_k = w) \ := \ P(A_k^w) \tag{1.8}$$

Definicja 1.3.10 *Niech Y będzie dowolnym zbiorem. Funkcję $f : \mathcal{I} \to Y$, spełniającą warunek:*

$$\forall\ x \in \mathcal{X} \ \ \forall\ i_1, i_2 \in \mathcal{I}_x \quad f(i_1) = f(i_2) \tag{1.9}$$

nazywamy warstwami stałą.

Warstwami stała funkcja f przyporządkowuje obiektom z tej samej warstwy dokładnie jeden element ze zbioru Y. Można powiedzieć, że f bezpośrednio działa na przykładach obiektów (jest uzależniona jedynie od zestawu atrybutów opisujących obiekt).

Twierdzenie 1.3.2 *Dla dowolnej funkcji $h : \mathcal{X} \to \mathbb{R}$ mierzalna jest funkcja:*

$$f : \mathcal{I} \to \mathbb{R} \quad \text{gdzie} \quad \mathcal{I} \ni i \mapsto h(x^i) \in \mathbb{R}$$

Dowód: Należy pokazać (def. A.4.1), że:

$$\forall\ a \in \mathbb{R} \quad \{ i \in \mathcal{I} \ : \ f(i) < a \} \in \mathfrak{B}$$

Funkcja f jest warstwami stała, tzn. dla każdego $x \in \mathcal{X}$ zachodzi:

$$\forall\ i_1, i_2 \in \mathcal{I}_x \quad f(i_1) = h(x^{i_1}) = h(x^{i_2}) = f(i_2)$$

$$\{i \in \mathcal{I}_x \ : \ f(i) < a\} = \mathcal{I}_x \quad \vee \quad \{i \in \mathcal{I}_x \ : \ f(i) < a\} = \emptyset$$

Z definicji rodziny $\mathcal{I}^{\mathcal{X}}$, wiemy, że $\bigcup_{x \in \mathcal{X}} \mathcal{I}_x = \mathcal{I}$. Istnieje więc $S \subseteq \mathcal{X}$, że:

$$\{i \in \mathcal{I} \ : \ f(i) < a\} = \bigcup_{x \in \mathcal{X}} \{i \in \mathcal{I}_x \ : \ f(i) < a\} = \bigcup_{x \in S} \mathcal{I}_x$$

Przypomnijmy, że $\mathcal{I}^{\mathcal{X}}$ jest rodziną zbiorów otwartych w przestrzeni obiektów (formalnie $\mathcal{I}^{\mathcal{X}} \subseteq \mathfrak{T}$). Zatem ze zbiorów otwartych składać się musi rodzina:

$$\{\mathcal{I}_x\}_{x \in S} \subseteq \mathcal{I}^{\mathcal{X}}$$

Unia dowolnej rodziny zbiorów otwartych jest zbiorem otwartym, zatem:

$$\{i \in \mathcal{I} \ : \ f(i) < a\} = \bigcup_{x \in S} \mathcal{I}_x \in \mathfrak{T} \subseteq \mathfrak{B}$$

Funkcja f jest więc mierzalna.

■

Wniosek 1.3.4 *Każda funkcja warstwami stała w $\mathcal{I}$, o wartościach ze zbioru $\mathbb{R}$ liczb rzeczywistych, jest mierzalna w przestrzeni obiektów $\mathfrak{I}$.*

Powyższe twierdzenie jest prawdziwe dzięki specjalnej konstrukcji topologii $\mathfrak{T}$ w $\mathcal{I}$, która zakłada, że warstwa każdego przykładu jest zbiorem otwartym.

1.3.4. Warstwa klasy i pojęcie docelowe

Niech będzie dany zbiór $C = \{1, \ldots, g\}$, gdzie $g \in \mathbb{N}$. Zbiór C nazywamy zbiorem klas (etykiet). W dalszej części każdemu obiektowi z $\mathfrak{I}$ przypiszemy dokładnie jedną klasę z C (przydzielimy go więc do pewnej populacji).

Definicja 1.3.11 *Funkcję mierzalną $\mathcal{E} : \mathcal{I} \to C$ spełniającą warunek:*

$$\mathcal{E}(\mathcal{I}) = C$$

nazywamy etykietą docelową przestrzeni obiektów $\mathfrak{I}$.

Zwróćmy uwagę, że etykieta docelowa działa bezpośrednio na zbiorze obiektów $\mathcal{I}$, klasyfikując obiekt $i \in \mathcal{I}$ do klasy $\mathcal{E}(i) \in C$. Posiadamy jedynie „indeks" $i \in \mathcal{I}$ obiektu, oraz wartości jego atrybutów (przykład) $x^i \in \mathcal{X}$. Zauważmy, że etykieta $\mathcal{E}$ pośrednio działa na przykładzie x^i przyporządkowanym do obiektu i (jej wartość może być uzależniona od przykładu). Ponadto możliwa jest sytuacja, gdzie różne obiekty opisane tym samym przykładem, klasyfikowane są do różnych klas lub nie.

Definicja 1.3.12 *Warstwą klasy $k \in C$ w przestrzeni obiektów $\mathfrak{I}$ nazywamy zbiór:*

$$\mathcal{I}^k := \{i \in \mathcal{I} \ : \ \mathcal{E}(i) = k\} \tag{1.10}$$

Warstwa klasy $k \in C$ jest zbiorem obiektów pochodzących z tej samej klasy k.

Definicja 1.3.13 *Warunkową warstwą klasy $k \in C$ (pod warunkiem, że znamy przykład $x \in \mathcal{X}$) w przestrzeni obiektów $\mathfrak{I}$, nazywamy zbiór:*

$$\mathcal{I}_x^k := \mathcal{I}^k \cap \mathcal{I}_x = \{i \in \mathcal{I}_x \ : \ \mathcal{E}(i) = k\} \tag{1.11}$$

Warunkowa warstwa klasy $k \in C$ (pod warunkiem, że znamy przykład $x \in \mathcal{X}$) jest zbiorem tych obiektów, pochodzących z tej samej klasy k, które opisuje wspólny przykład x.

Wniosek 1.3.5 *Zbiory $\mathcal{I}^k$ i $\mathcal{I}_x^k$ są mierzalne, formalnie $\mathcal{I}^k, \mathcal{I}_x^k \in \mathfrak{B}$. Zachodzą ponadto równości:*

$$\bigcup_{x \in \mathcal{X}} \mathcal{I}_x = \mathcal{I} \tag{1.12}$$

$$\bigcup_{k \in C} \mathcal{I}^k = \mathcal{I} \tag{1.13}$$

$$\bigcup_{x \in \mathcal{X}} \mathcal{I}_x^k = \mathcal{I}^k \tag{1.14}$$

$$\bigcup_{k \in C} \mathcal{I}_x^k = \mathcal{I}_x \tag{1.15}$$

Warstwa klasy jest mierzalna ze względu na mierzalność etykiety docelowej $\mathcal{E}$ (def. A.4.1, tw. A.4.2, własność w_3). Mierzalność warunkowej warstwy klasy wynika z mierzalności przecięcia dwóch zbiorów mierzalnych (def. A.2.1, tw. A.2.1, własność w_4).

Definicja 1.3.14 *Każdą etykietę docelową $\mathcal{E} : \mathcal{I} \to C$, dla której zachodzi:*

$$\forall \ k \in C \quad P(\mathcal{I}^k) > 0$$

nazywamy pojęciem docelowym w przestrzeni obiektów $\mathfrak{I}$.

1.3.5. Rozkłady *a priori* i *a posteriori*

Zauważmy, że pojęcie docelowe wprowadza w zbiorze przykładów trzy rozkłady, zwane dalej prawdopodobieństwami *a priori*[3] i *a posteriori*[4]:

Definicja 1.3.15 *Rozkładami a priori wprowadzonymi przez pojęcie docelowe nazywamy:*

1. $\pi_k := P(\mathcal{I}^k)$ *- prawdopodobieństwo klasy k,*
2. $p(x|k) := P(\mathcal{I}_x|\mathcal{I}^k) = \dfrac{P(\mathcal{I}_x \cap \mathcal{I}^k)}{P(\mathcal{I}^k)} = \dfrac{P(\mathcal{I}_x^k)}{P(\mathcal{I}^k)}$ *- prawdopodobieństwo że x pod warunkiem, że klasa k.*

Definicja 1.3.16 *Rozkładem a posteriori wprowadzonym przez pojęcie docelowe nazywamy prawdopodobieństwo, że ustalony przykład x pochodzi z klasy k:*

$$p(k|x) := P(\mathcal{I}^k|\mathcal{I}_x) = \frac{P(\mathcal{I}^k \cap \mathcal{I}_x)}{P(\mathcal{I}_x)} = \frac{P(\mathcal{I}_x^k)}{P(\mathcal{I}_x)}$$

[3] W języku łacińskim - z założenia.
[4] W języku łacińskim - z następstwa.

Stwierdzenie 1.3.1 *Zachodzą równości:*

$$\sum_{k\in C} \pi_k = 1 \tag{1.16}$$

$$\forall\ k \in C \ \sum_{x\in\mathcal{X}} p(x|k) = 1 \tag{1.17}$$

$$\forall\ x \in X \ \sum_{k\in C} p(k|x) = 1 \tag{1.18}$$

Dowód: Wykorzystując podstawowe własności miary probabilistycznej:

(1.16)

$$\sum_{k\in C} \pi_k = \sum_{k\in C} P(\mathcal{I}^k) = P\left(\bigcup_{k\in C} \mathcal{I}^k\right) = P(\mathcal{I}) = 1$$

(1.17) ustalmy $k \in C$

$$\sum_{x\in\mathcal{X}} p(x|k) = \sum_{x\in\mathcal{X}} \frac{P(\mathcal{I}^k_x)}{P(\mathcal{I}^k)} = \frac{\sum_{x\in\mathcal{X}} P(\mathcal{I}^k_x)}{P(\mathcal{I}^k)} = \frac{P(\bigcup_{x\in\mathcal{X}} \mathcal{I}^k_x)}{P(\mathcal{I}^k)} = \frac{P(\mathcal{I}^k)}{P(\mathcal{I}^k)} = 1$$

(1.18) analogicznie do (1.17).

■

Stwierdzenie 1.3.2 *Z twierdzenia Bayesa (A.3.4) wynika, że:*

$$p(k|x) = \frac{\pi_k p(x|k)}{\sum_{r\in C} \pi_r p(x|r)} \tag{1.19}$$

Rozkład a posteriori jest więc jednoznacznie wyznaczony przez rozkłady a priori. Jest to bardzo ważny wniosek, który w dalszej części pozwoli na konstrukcję pewnej specjalnej rodziny klasyfikatorów[5].

1.3.6. Przestrzeń ucząca

Przypomnijmy, że w przestrzeni obiektów $\mathfrak{I}$ z pojęciem docelowym $\mathcal{E}$ każdy obiekt $i \in \mathcal{I}$ pochodzi z klasy $\mathcal{E}(i) \in C$. W praktyce nigdy nie znamy całej przestrzeni obiektów. Najczęściej posiadamy „skończoną" informację, która składa się z podrodziny przykładów opisujących pewne obiekty (ogólnie nieznane), wraz z listą klas, do których te obiekty przynależą. Można powiedzieć, że dane jest obcięcie pojęcia docelowego do pewnego podzbioru zbioru obiektów, co w praktycznym sensie odzwierciedla jedynie pośrednią zależność pojęcia docelowego od przykładu. Dalej skoncentrujemy się na uogólnieniu przedstawionego powyżej, intuicyjnego pojęcia, określanego mianem *zbioru uczącego.*

Definicja 1.3.17 *Niech będzie dana przestrzeń obiektów $\mathfrak{I} = (\mathcal{I}, \mathfrak{B}, P)$ wraz z pojęciem docelowym $\mathcal{E} : \mathcal{I} \to C$. Każdą parę $(S, \mathcal{E}_S)$ gdzie $S \in \mathfrak{B}$ i $P(S) > 0$, a $\mathcal{E}_S : S \to C$ jest funkcją daną wzorem $\mathcal{E}_S := \mathcal{E}|_S$ (obcięcie $\mathcal{E}$ do S) nazywamy zbiorem uczącym w przestrzeni obiektów $\mathfrak{I}$. Funkcję $\mathcal{E}_S$ nazywamy pojęciem indukowanym do zbioru S.*

[5]Rodzina klasyfikatorów bayesowskich przy znanych estymatorach prawdopodobieństw a priori.

Idea klasyfikacji opiera się na tym, aby na podstawie podanego zbioru uczącego, możliwe było uogólnienie (rozszerzenie) pojęcia indukowanego do etykiety jak najbliższej tej, która reprezentuje pojęcie docelowe.

Definicja 1.3.18 *Rodzinę zbiorów uczących:*

$$\mathcal{L}(\mathfrak{I},\mathcal{E}) = \Big\{(S,\ \mathcal{E}_S)\ :\ S \in \mathfrak{B},\ P(S) > 0\Big\} \tag{1.20}$$

nazywamy przestrzenią uczącą w przestrzeni obiektów $\mathfrak{I}$ z pojęciem docelowym $\mathcal{E}$.

W dalszych rozważaniach przestrzeń uczącą będziemy oznaczamy tylko symbolem $\mathcal{L}$.

1.3.7. Reguła dyskryminacyjna i przestrzeń hipotez

Wspomnieliśmy wcześniej o idei uogólnienia (rozszerzenia na podstawie zbioru uczącego) pojęcia indukowanego do etykiety jak najbliższej pojęciu docelowemu. Poniżej podamy definicję, która w naturalny sposób wprowadza takie uogólnienie.

Definicja 1.3.19 *Regułą dyskryminacyjną (klasyfikatorem) nazywamy każdą funkcję*

$$d : \mathcal{X} \times \mathcal{L} \to C, \quad \text{gdzie} \quad \mathcal{X} \times \mathcal{L} \ni (x,\mathcal{S}) \mapsto d(x,\mathcal{S}) \in C \tag{1.21}$$

Definicja 1.3.20 *Funkcję*

$$d_{\mathfrak{I}} : \mathcal{I} \times \mathcal{L} \to C, \quad \text{gdzie} \quad \mathcal{I} \times \mathcal{L} \ni (i,\mathcal{S}) \mapsto d(x^i,\mathcal{S}) \in C \tag{1.22}$$

nazywamy przedłużeniem klasyfikatora d ze zbioru przykładów $\mathcal{X}$ na przestrzeń obiektów $\mathfrak{I}$.

Reguła dyskryminacyjna działa (pod warunkiem ustalenia zbioru uczącego) na zbiorze przykładów (nie obiektów). Czyli dla dowolnego $\mathcal{S} \in \mathcal{L}$ mamy:

$$d(\ \cdot\ ,\mathcal{S}) : \mathcal{X} \to C$$

Można powiedzieć, że pod warunkiem podania zbioru uczącego $\mathcal{S}$ klasyfikator d klasyfikuje przykład $x \in \mathcal{X}$ do klasy $d(x,\mathcal{S}) \in C$ (mówimy wtedy, że zbiór uczący $\mathcal{S}$ uczy klasyfikator d). Klasyfikator d potrafi klasyfikować obiekty (działać na obiektach) poprzez swoje przedłużenie $d_{\mathfrak{I}}$ na przestrzeń obiektów $\mathfrak{I}$.

Stwierdzenie 1.3.3 *Dla dowolnego zbioru uczącego $\mathcal{S} \in \mathcal{L}$ funkcja:*

$$d_{\mathfrak{I}}(\ \cdot\ ,\mathcal{S}) : \mathcal{I} \to C$$

jest funkcją mierzalną.

Uzasadniając powyższe stwierdzenie, wystarczy zauważyć, że przedłużenie $d_{\mathfrak{I}}$ klasyfikatora d jest funkcją warstwami stałą (def. 1.3.10, tw. 1.3.2) w $\mathcal{I}$ (pod warunkiem ustalenia zbioru uczącego $\mathcal{S}$).

Definicja 1.3.21 *Przestrzenią hipotez nazywamy zbiór*

$$\mathcal{H}(\mathcal{X},C) := \Big\{d(\ \cdot\ ,\mathcal{S}) : \mathcal{X} \to C\ :\ \mathcal{S} \in \mathcal{L},\ d - \text{klasyfikator}\Big\} \tag{1.23}$$

Oznaczenie $\mathcal{H}(\mathcal{X}, C)$ sugeruje utratę bezpośredniego związku pomiędzy pojęciem hipotezy a przestrzenią obiektów. W dalszej części zamiast $\mathcal{H}(\mathcal{X}, C)$ pisać będziemy $\mathcal{H}$

Stwierdzenie 1.3.4 *Zachodzi równość:* $\mathcal{H}(\mathcal{X}, C) = \mathcal{X}^C$

Dowód: Z definicji przestrzeni hipotez bezpośrednio wynika, że $\mathcal{H}(\mathcal{X}, C) \subseteq \mathcal{X}^C$. Aby pokazać, że $\mathcal{X}^C \subseteq \mathcal{H}(\mathcal{X}, C)$ weźmy dowolną funkcję $h \in \mathcal{X}^C$. Dla każdego $x \in \mathcal{X}$, $S \in \mathcal{L}$ określamy:

$$d(x, \mathcal{S}) := h(x)$$

Oczywiście $d : \mathcal{X} \times \mathcal{L} \to C$ jest klasyfikatorem $\Rightarrow h \in \mathcal{H}(\mathcal{X}, C)$.

■

Definicja 1.3.22 *Przedłużeniem hipotezy* $h \in \mathcal{H}$ *na przestrzeń obiektów* $\mathfrak{I}$ *nazywamy funkcję:*

$$h_{\mathfrak{I}} : \mathcal{I} \to C, \quad \text{gdzie } \mathcal{I} \ni i \mapsto h(x^i) \in C \tag{1.24}$$

Każda hipoteza klasyfikuje przykłady. Dla każdej hipotezy istnieje klasyfikator (wraz ze zbiorem uczącym) ją generujący. Hipoteza poprzez swoje przedłużenie na przestrzeń obiektów klasyfikuje również obiekty.

Wniosek 1.3.6 *Przedłużenie* $h_{\mathfrak{I}}$ *hipotezy* h *jest funkcją mierzalną.*

Powyższy wniosek wynika z twierdzenia 1.3.2. Przedłużenie hipotezy klasyfikuje obiekty opisane tym samym przykładem do wspólnej klasy (jest funkcją warstwami stałą).

1.3.8. Odległość w przestrzeni hipotez

Mając zebrane wszystkie możliwe hipotezy w jednym miejscu chcielibyśmy móc ocenić ich „jakość" (inaczej miarę błędnych klasyfikacji). W tym celu wprowadzimy odległość w przestrzeni hipotez, która pozwoli porównać dwie dowolne hipotezy. Podamy również definicję odległości hipotezy od pojęcia docelowego.

Definicja 1.3.23 *Niech* $g, h \in \mathcal{H}$ *oznacza dwie dowolne hipotezy. Różnicą hipotez* g *i* h *nazywamy zbiór:*

$$g \setminus h := \left\{ i \in \mathcal{I} \ : \ g_{\mathfrak{I}}(i) \neq h_{\mathfrak{I}}(i) \right\} \tag{1.25}$$

gdzie $g_{\mathfrak{I}}, h_{\mathfrak{I}}$ *są przedłużeniami hipotez* g, h *na przestrzeń obiektów* $\mathfrak{I}$*.*

Stwierdzenie 1.3.5 $g \setminus h \in \mathfrak{B}$ *dla dowolnych hipotez* $g, h \in \mathcal{H}$*.*

Przedłużenie dowolnej hipotezy jest funkcją mierzalną w $\mathfrak{I}$. Uzasadnienie kończymy podając twierdzenia A.4.2, własność (w_4).

Definicja 1.3.24 *Odległością hipotez* $g, h \in \mathcal{H}$ *nazywamy prawdopodobieństwo:*

$$P(g \setminus h)$$

Tak zdefiniowana odległość ma kilka naturalnych własności:

1. $P(h \setminus h) = 0$ dla każdej hipotezy $h \in \mathcal{H}$
2. $P(g \setminus h) = P(h \setminus g)$ dla dowolnych $g, h \in \mathcal{H}$

3. odległość nieprzecinających się hipotez wynosi 1.

Definicja 1.3.25 *Różnicą dowolnej hipotezy $h \in \mathcal{H}$ i pojęcia docelowego $\mathcal{E}$ nazywamy zbiór:*

$$\mathcal{E} \setminus h := \Big\{ i \in \mathcal{I} \ : \ \mathcal{E}(i) \neq h_{\mathfrak{I}}(i) \Big\} \tag{1.26}$$

Na podstawie mierzalności funkcji $\mathcal{E}$ i mierzalności przedłużenia $h_{\mathfrak{I}}$ hipotezy h stwierdzamy, że $\mathcal{E} \setminus h \in \mathfrak{B}$. Niech będzie dany przykład $x \in \mathcal{X}$. $\mathcal{I}_x$ jest zbiorem obiektów opisanych przykładem x. Przedłużenie $h_{\mathfrak{I}}$ hipotezy h klasyfikuje wszystkie obiekty z warstwy $\mathcal{I}_x$ do tej samej klasy $h(x)$. Łatwo zatem o wniosek, że warunkowa warstwa $\mathcal{I}_x^{h(x)}$ nie jest obarczona błędem klasyfikacji, gdzie poza nią błąd jest z pewnością popełniany. Istotnie:

$$\mathcal{I}_x^{h(x)} = \Big\{ i \in \mathcal{I}_x \ : \ \mathcal{E}(i) = h(x) \Big\}$$

Wniosek 1.3.7 *Zachodzą równości:*

$$\mathcal{E} \setminus h = \mathcal{I} \setminus \bigcup_{x \in \mathcal{X}} \mathcal{I}_x^{h(x)} = \bigcup_{x \in \mathcal{X}} \mathcal{I}_x \setminus \mathcal{I}_x^{h(x)}$$

Otrzymaliśmy dość klarowny obraz struktury zbioru będącego różnicą pojęcia docelowego i dowolnej hipotezy.

Definicja 1.3.26 *Błędem rzeczywistym hipotezy $h \in \mathcal{H}$ nazywamy prawdopodobieństwo:*

$$P(\mathcal{E} \setminus h)$$

Można powiedzieć, że błąd rzeczywisty hipotezy h reprezentuje jej odległość od pojęcia docelowego.

Definicja 1.3.27 *Błędem indukowanym (błędem próby) hipotezy $h \in \mathcal{H}$ na zbiorze uczącym $(S, \mathcal{E}_S) \in \mathcal{L}$ nazywamy prawdopodobieństwo warunkowe:*

$$P\Big(\mathcal{E} \setminus h \mid S\Big) = P\bigg(\Big\{ i \in S \ : \ \mathcal{E}_S(i) \neq h_{\mathfrak{I}}(i) \Big\}\bigg)$$

1.3.9. Problem dyskryminacyjny

Wyznaczenie błędu rzeczywistego hipotezy najczęściej w praktyce nie jest możliwe (pojęcie docelowe jest na ogół nieznane). Posiadamy jedynie zbiór uczący będący podzbiorem zbioru obiektów wraz z pojęciem indukowanym do tego zbioru. Pozwala to na estymację błędu rzeczywistego hipotezy i często na dokładne wyznaczenie jej błędu indukowanego.

Definicja 1.3.28 *Zadanie wyboru klasyfikatora d nazywamy problemem dyskryminacyjnym w przestrzeni obiektów $\mathfrak{I}$ ze zbiorem uczącym S.*

Idea klasyfikacji sprowadza się do poszukiwania hipotezy jak najbardziej zbliżonej do pojęcia docelowego. Dalej skoncentrujemy się na podaniu hipotezy minimalizującej odległość od pojęcia docelowego.

$$P(\mathcal{E} \setminus h) \underset{h \in \mathcal{H}}{\longrightarrow} \min \tag{1.27}$$

1.4. Reguła Bayesa

W praktyce rozkłady a priori (wyznaczone przez pojęcie docelowe) nie są znane. Posiadamy jedynie informacje o pojęciu indukowanym do zbioru uczącego, co pozwala na estymację. Poniżej przedstawiamy klasyfikator, którego konstrukcja umożliwia wykorzystanie estymatorów prawdopodobieństw a priori.

1.4.1. Klasyfikator bayesowski

Przy znanym rozkładzie a posteriori $p(k|x)$ najbardziej naturalną jest hipoteza, która klasyfikuje przykład x do klasy k z maksymalnym prawdopodobieństwem $p(k|x)$ (def. 1.3.16).

Definicja 1.4.1 *Funkcję $\mathcal{E}^{\mathfrak{b}} : \mathcal{X} \to C$ daną wzorem:*

$$\mathcal{E}^{\mathfrak{b}}(x) := \arg\max_k p(k|x) \tag{1.28}$$

nazywamy etykietą bayesowską. W przypadku istnienia kilku klas z maksymalnym prawdopodobieństwem a posteriori, etykieta bayesowska wybiera jedną z nich (dowolną).

Etykieta bayesowska jest oczywiście hipotezą. Formalnie: $\mathcal{E}^{\mathfrak{b}} \in \mathcal{H}$.

Wniosek 1.4.1 *Z maksymalizacji prawdopodobieństwa a posteriori $p(k|x)$ wynika:*

$$\forall\ x \in \mathcal{X}\ \ \forall\ k \in C \quad P\Big(\mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\Big) \geqslant P(\mathcal{I}_x^k)$$

Wniosek 1.4.2 *Wybór takiej klasy k, że maksymalne jest prawdopodobieństwo $p(k|x)$, równoważny jest wyborowi takiego k, że maksymalna jest wartość wyrażenia $\pi_k p(x|k)$ (stw. 1.3.2). Ogólnie zachodzi:*

$$\mathcal{E}^{\mathfrak{b}}(x) := \arg\max_k \pi_k p(x|k) \tag{1.29}$$

Hipoteza to inaczej „nauczony klasyfikator". W przypadku reguły bayesowskiej zbiór uczący wykorzystywany jest do estymacji rozkładów a priori (wspomnieliśmy wcześniej, że rozkłady te w praktyce nie są znane).

1.4.2. Optymalność reguły bayesowskiej

Poniżej formułujemy twierdzenie, które można nazwać fundamentalnym w analizie dyskryminacyjnej.

Twierdzenie 1.4.1 *Etykieta bayesowska minimalizuje odległość od pojęcia docelowego.*

Dowód: Należy pokazać, że:

$$\forall\ h \in \mathcal{H} \quad P(\mathcal{E} \setminus h) \geqslant P(\mathcal{E} \setminus \mathcal{E}^b)$$

Ustalmy więc $h \in \mathcal{H}$. Przywołując wniosek 1.3.7 zapisujemy:

$$\mathcal{E} \setminus h = \bigcup_{x \in \mathcal{X}} \mathcal{I}_x \setminus \mathcal{I}_x^{h(x)} \qquad \mathcal{E} \setminus \mathcal{E}^{\mathfrak{b}} = \bigcup_{x \in \mathcal{X}} \mathcal{I}_x \setminus \mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}$$

Z przeliczalności zbioru $\mathcal{X}$ (tw. 1.3.1) i przeliczalnej addytywności miary (def. A.3.1, własność μ_3) otrzymujemy:

$$P(\mathcal{E} \setminus h) = \sum_{x \in \mathcal{X}} P\Big(\mathcal{I}_x \setminus \mathcal{I}_x^{h(x)}\Big) = \sum_{x \in \mathcal{X}} \Big(P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{h(x)}\big)\Big)$$

$$P(\mathcal{E} \setminus \mathcal{E}^{\mathfrak{b}}) = \sum_{x \in \mathcal{X}} P\Big(\mathcal{I}_x \setminus \mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\Big) = \sum_{x \in \mathcal{X}} \Big(P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\big)\Big)$$

Wniosek 1.4.1 stwierdza bezpośrednio:

$$P\big(\mathcal{I}_x^{h(x)}\big) \leqslant P\big(\mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\big)$$

Zatem:

$$P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{h(x)}\big) \geqslant P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\big)$$

$$\sum_{x \in \mathcal{X}} \Big(P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{h(x)}\big)\Big) \geqslant \sum_{x \in \mathcal{X}} \Big(P(\mathcal{I}_x) - P\big(\mathcal{I}_x^{\mathcal{E}^{\mathfrak{b}}(x)}\big)\Big)$$

Otrzymujemy więc tezę:

$$P(\mathcal{E} \setminus h) \geqslant P(\mathcal{E} \setminus \mathcal{E}^{b})$$

■

Pokazaliśmy istnienie jednoznacznie wyznaczonej „najlepszej" hipotezy (hipotezy z najmniejszym prawdopodobieństwem błędu). Jest nią etykieta bayesowska $\mathcal{E}^{\mathfrak{b}}$. Wskazaliśmy ponadto, że etykieta bayesowska może być przybliżana hipotezami bayesowskimi w zależności od jakości estymacji rozkładów a priori.

Rozdział 2

Drzewa klasyfikacyjne

2.1. Wprowadzenie

Drzewa klasyfikacyjne (*decyzyjne*) pojawiły się niezależnie w nauczaniu maszynowym i w statystyce. Oparte na nich algorytmy są najczęściej wykorzystywane. Struktura drzew decyzyjnych pozwala na konstrukcję najogólniejszych reguł klasyfikacyjnych, efektywnych w implementacji i przejrzystych w logicznej konstrukcji. Na szczególną uwagę zasługuje przydatność struktur do rozwiązywania zadań o dużym *wymiarze prób losowych*[1] i/lub lub dużym *wymiarze wektora obserwacji*[2] (np.: klasyfikacja kredytobiorców, predykcja predyspozycji klienta do odejścia).

Podstawową wielkością charakteryzującą „dobroć" algorytmu jest jego *złożoność obliczeniowa*, która pokazuje zależność pomiędzy *czasem*[3] działania algorytmu, a jego parametrami wejściowymi. Definiuje się również pojęcie *złożoności pamięciowej*, jednak istnienie nośników potrafiących pomieścić TB[4] danych zmniejsza wagę tej wielkości.

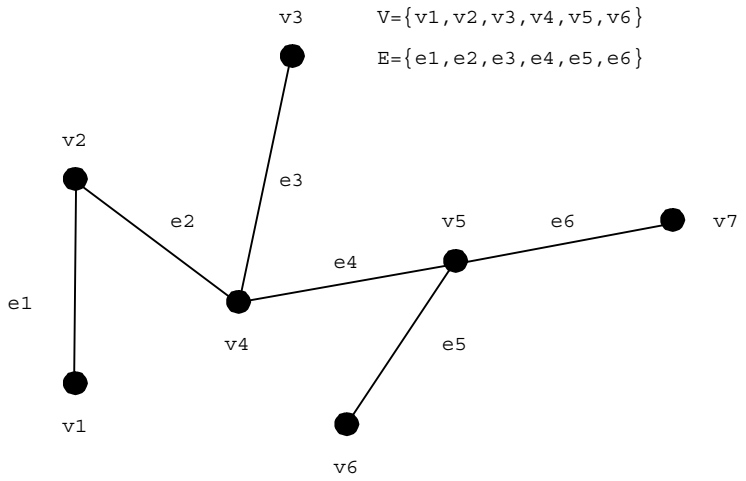
Skalowalność procesu to „zawieranie" się w nim harmonijnych zależności pomiędzy sposobem jego działania i zmianą warunków początkowych. Dla algorytmów jest to np. proporcjonalność czasu działania do wielkości danych wejściowych. W terminologii systemów informatycznych skalowalność definicjainiuje się jako możliwość harmonijnego rozrastania się systemu w miarę upływu czasu i zwiększania liczby jego użytkowników, bez konieczności rewolucyjnych zmian projektowych.

Poniższy tekst wprowadza formalne definicje, zakładając istnienie przestrzeni obiektów $\mathfrak{I} = (\mathcal{I}, \mathfrak{B}, P)$ z rodziną obserwacji $\mathcal{X}^{\mathcal{I}} = \{x^i\}_{i\in\mathcal{I}}$, zbiorem przykładów $\mathcal{X}$ rodziny $\mathcal{X}^{\mathcal{I}}$ oraz pojęciem docelowym $\mathcal{E} : \mathcal{I} \to C$, gdzie $C = \{1, ..., g\}$ $g \in \mathbb{N}$ jest zbiorem etykiet klas (definicje: 1.3.4, 1.3.5, 1.3.8, 1.3.14).

2.2. Struktura drzewa

W teorii grafów drzewem (def. B.2.1) nazywamy dowolny graf (def. B.1.1) spójny (def. B.1.10) i acykliczny (def. B.1.9) . Rozpatrzmy drzewo $T =< V, E >$ o zbiorze wierzchołków

[1] Najczęściej liczba obserwacji w zbiorze uczącym / testowym.
[2] Liczba atrybutów opisujących obserwacje.
[3] W ogólności liczba iteracji.
[4] Terabajt [TB] = 2^{30} bajtów



Rysunek 2.1: Graf będący drzewem

V i krawędzi E. W zbiorze V wyróżniamy podzbiór wierzchołków $L^T \subset V$ będących liśćmi (def. B.2.2) drzewa T. Wykorzystując pojęcie stopnia wierzchołka (def. B.1.5) zapisujemy:

$$L^T := \{v \in V \ : \ \deg_T(v) = 1\}$$

Ustalmy wierzchołek $r \in V$ drzewa T i nazwijmy go *korzeniem* drzewa T. Oznaczmy przez L_r^T zbiór:

$$L_r^T := L^T \setminus \{r\} \tag{2.1}$$

W szczególnym przypadku korzeń r może być liściem drzewa T. Zbiór L_r^T nie zawiera wtedy korzenia r.

Definicja 2.2.1 *Zbiór*

$$N_r^T := V \setminus L_r^T \tag{2.2}$$

nazywamy zbiorem węzłów drzewa T z ustalonym korzeniem r.

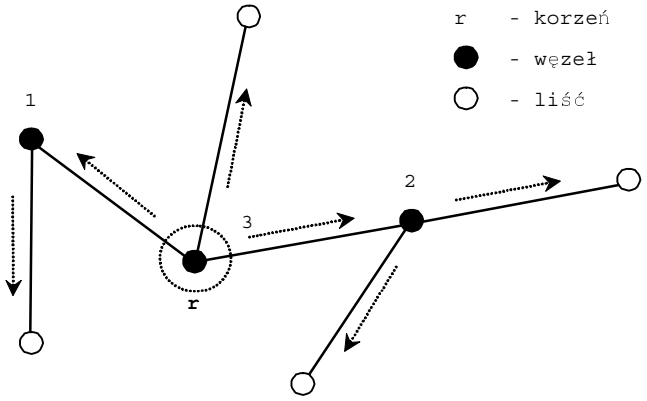
Do zbioru węzłów drzewa T zaliczają się wszystkie wierzchołki o stopniu wyższym niż 1 oraz ustalony korzeń $r \in V$.

Dla dowolnych wierzchołków $u, v \in V$ drzewa T istnieje dokładnie jedna $u - v$ droga (def. B.1.6) i jest to droga prosta (def. B.1.7, tw. B.2.1). W szczególności, dla dowolnego liścia $l \in L_r^T$ istnieje dokładnie jedna $r - l$ droga prosta łącząca korzeń r z liściem l. Mówimy, że $r - l$ droga prowadzi od korzenia r, przez węzły, do liścia l.

Definicja 2.2.2 *Liczbę*

$$\text{split}_{T_r}(n) := \begin{cases} \deg_T(n) - 1 & \textit{jeśli } n \neq r \\ \deg_T(n) & \textit{jeśli } n = r \end{cases} \tag{2.3}$$

nazywamy współczynnikiem rozgałęzienia w węźle $n \in N_r^T$ drzewa T z ustalonym korzeniem r.



Rysunek 2.2: Graf będący drzewem z wyróżnionym korzeniem

Każdy węzeł $n \in N_r^T$ posiada dokładnie $\deg_T(n)$ wierzchołków sąsiadujących (def. B.1.4) . W przypadku $n \neq r$ istnieje dokładnie jeden węzeł s sąsiadujący z n, który leży na $r-n$ drodze (od korzenia r do węzła n). Pozostałe wierzchołki sąsiadujące z n leżą na drogach od korzenia, do liścia, przechodzących przez węzeł n. Traktując rzecz obrazowo powiemy, że droga od korzenia r, przez węzeł n, prowadząc do liścia, może w węźle n rozgałęzić się na $\mathrm{split}_{T_r}(n)$ sposobów (rysunek 2.2).

Definicja 2.2.3 *Poprzednikiem (węzłem macierzystym) wierzchołka $n \in V$ różnego od korzenia r, nazywamy węzeł s sąsiadujący z n leżący na $r-n$ drodze. Piszemy wtedy $s \succ n$. Przyjmujemy, że korzeń r nie posiada poprzedników.*

Każdy wierzchołek różny od korzenia posiada dokładnie jeden węzeł macierzysty.

Definicja 2.2.4 *Następnikiem (potomkiem) węzła $n \in N_r^T$ nazywamy każdy wierzchołek m (węzeł lub liść) sąsiadujący z n i nie będący jego poprzednikiem. Piszemy wtedy $n \succ m$. Zbiór*

$$n_{\succ} := \Big\{ m \in V \ : \ n \succ m \Big\} \tag{2.4}$$

nazywamy zbiorem następników węzła n.

Wniosek 2.2.1 *Zachodzi równość*

$$|n_{\succ}| = \mathrm{split}_{T_r}(n)$$

Definicja 2.2.5 *Testem w węźle $n \in N_r^T$ nazywamy każdą funkcję:*

$$t_n : X \to n_{\succ} \quad \text{gdzie} \quad X \ni x \mapsto n_x \in n_{\succ} \tag{2.5}$$

Zauważmy, że test jest funkcją przyporządkowaną do węzła, która przeprowadza przykłady na następniki tego węzła.

Definicja 2.2.6 *Mówimy, że test $t_n : X \to n_{\succ}$ w węźle $n \in N_r^T$ jest testem atrybutu A_k, $k \in \{1, \ldots, p\}$ jeżeli istnieje taka funkcja $t_n^k : X_k \to n_{\succ}$, że:*

$$\forall\ x \in X \quad t_n(x) = t_n^k(A_k(x)) \tag{2.6}$$

Definicja 2.2.7 *Przekształcenie:*

$$c : L_r^T \to C \quad \text{gdzie} \quad L_r^T \ni l \mapsto k_l \in C \tag{2.7}$$

nazywamy etykietą liści $l \in L_r^T$ drzewa T.

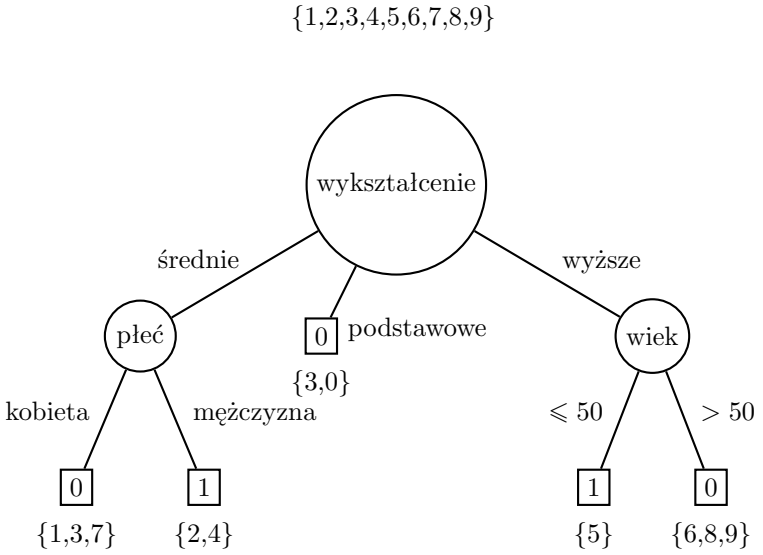
Definicja 2.2.8 *Drzewem klasyfikacyjnym (decyzyjnym) nazywamy każde drzewo $T_r =< V, E >$ z korzeniem $r \in V$, rodziną testów $\{t_n\}_{n \in N_r^T}$ oraz etykietą liści $c : L_r^T \to C$. Zbiór L_r^T nazywamy zbiorem liści drzewa klasyfikacyjnego T_r. Zbiór N_r^T nazywamy zbiorem węzłów drzewa klasyfikacyjnego T_r.*

Definicja 2.2.9 *Mówimy, że drzewo klasyfikacyjne $T_r =< V, E >$ jest drzewem binarnym, jeżeli:*

$$\forall\ n \in N_r^T \quad |n_{\succ}| = 2$$

Drzewo klasyfikacyjne jest drzewem, które posiada dodatkową interpretację dla węzłów, gałęzi i liści:

- węzły odpowiadają testom przeprowadzanym na wartościach atrybutów przykładów, węzeł drzewa, który nie ma żadnych węzłów macierzystych jest *korzeniem*,
- gałęzie odpowiadają możliwym wynikom tych testów,
- liście odpowiadają etykietom klas danego problemu dyskryminacji (w konwencji drzewo klasyfikacyjne ma więcej niż 1 liść),
- drzewo „rośnie" od góry do dołu (od korzenia do liści).

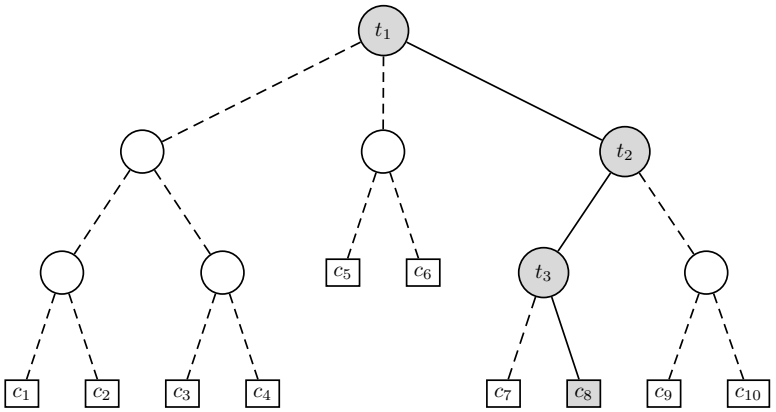


Rysunek 2.3: Przykład drzewa klasyfikacyjnego

Zaobserwowane elementy próby przesuwają się wzdłuż gałęzi przez węzły. W węzłach podejmowane są decyzje o wyborze gałęzi, wzdłuż której trwa przesuwanie. W każdym węźle mamy do czynienia z podziałem elementów do niego docierających na podgrupy (względem zapisanego w nim *kryterium podziału - testu*). Przesuwanie trwa do momentu, gdy napotkamy liść,

który ma etykietę którejś z klas. Rysunek 2.3 przedstawia przykład drzewa klasyfikacyjnego.

Dla każdego liścia istnieje dokładnie jedna droga łącząca go z korzeniem. Zbiór wszystkich takich dróg może być przekształcony do *zbioru reguł* (na ogół koniunkcji pewnych warunków elementarnych), klasyfikujących przykłady w sposób identyczny jak „robi" to drzewo. Możliwa jest więc konwersja drzewa decyzyjnego do zbioru reguł. Ze względu na czytelność i pamięciową oszczędność reprezentacji nie zawsze jest to uzasadnione działanie. Konwersja wykorzystywana jest przy *przycinaniu* drzewa, czyli zapobieganiu *nadmiernemu dopasowaniu* (2.5). Przykładowa ścieżka (droga pomiędzy korzeniem i liściem) w drzewie klasyfikacyjnym została przedstawiona na rysunku 2.4 - od korzenia → przez węzły i gałęzie → do liścia.



Rysunek 2.4: Ścieżka w drzewie klasyfikacyjnym

2.3. Drzewo jako hipoteza

Przedstawiliśmy obrazowo sposób klasyfikacji przykładów przez drzewo decyzyjne. Poniżej podamy definicję formalną funkcji klasyfikującej stowarzyszonej z drzewem klasyfikacyjnym.

Niech będzie dane drzewo klasyfikacyjne $T_r =< V, E >$ z korzeniem r, rodziną testów $\{t_n\}_{n \in N_r^T}$ i etykietą liści c.

Definicja 2.3.1 *Hipotezą h^T reprezentowaną drzewem klasyfikacyjnym T nazywamy przekształcenie zdefiniowane regułą rekurencyjną:*

1. *ustalamy $x \in \mathcal{X}$, $n_0 = r$*
2. *$n_{i+1} := t_{n_i}(x)$ - wykonuj działanie dopóki wynik nie będzie liściem,*
3. *jeżeli w k-tym kroku $n_k \in L_r^T$ (jest liściem), to zwróć etykietę liścia $c(n_k) \in C$.*

Twierdzenie 2.3.1 *Funkcja h^T jest hipotezą (def. 1.3.21) .*

Dowód: Z definicji 2.3.1 wynika, że $h^T \in \mathcal{X}^C$, co kończy dowód na mocy stwierdzenia 1.3.4. ■

Wniosek 2.3.1 *W drzewie klasyfikacyjnym T istnieje dokładnie jeden liść l_x związany z przykładem $x \in \mathcal{X}$ określony rekursją w krokach* 1 *i* 2 *definicji 2.3.1.*

Błąd rzeczywisty (indukowany / próby) hipotezy reprezentowanej przez drzewo nazywać będziemy błędem rzeczywistym (indukowanym / próby) drzewa.

2.4. Metody konstrukcji drzew

Pokazaliśmy, że drzewa klasyfikacyjne reprezentują hipotezy. Przypomnijmy, że hipoteza jest wynikiem uczenia się klasyfikatora (def. 1.3.19) . W praktyce najczęściej zachodzi konieczność utworzenia drzewa decyzyjnego dedykowanego do danego problemu dyskryminacyjnego. Poniżej przedstawimy podstawowe metody konstrukcji drzew reprezentujących hipotezy przybliżające pojęcia docelowe na podstawie dostępnych zbiorów uczących (def. 1.3.17) . Rozszerzymy tym samym pojęcie drzewa decyzyjnego do klasyfikatora.

Naszym celem jest zbudowanie drzewa klasyfikacyjnego z możliwie małym błędem rzeczywistym i małym błędem indukowanym. W praktyce stosuje się estymację błędu rzeczywistego (pojęcie docelowe jest nieznane). Minimalizacja obu błędów jednocześnie nie jest na ogół możliwa. Często dochodzi do sytuacji, w której na rzecz mniejszego błędu rzeczywistego pozwala się na większy błąd próby. W zadaniu budowy drzewa decyzyjnego wyróżnia się cztery podstawowe składowe:

1. Rodzinę $\{t_n^s\}$ testów określających podział w każdym węźle.
2. Zdefiniowane kryterium $\varphi(t_n^s)$ jakości podziału określone dla każdego testu t_n^s, w każdym węźle n.
3. Kryterium stopu budowy drzewa.
4. Konstrukcja reguły decyzyjnej (etykiety liści drzewa).

2.4.1. Konstrukcja testów

Dobór odpowiedniego testu jest decyzją o kluczowym znaczeniu dla późniejszych właściwości drzewa. Test powinien zapewniać możliwie dokładną klasyfikację dostępnych przykładów. Konstrukcja testów jest wysoce uzależniona od typu testowanego atrybutu. Przedstawimy jedynie testy binarne[5] zależne od wartości pojedynczych atrybutów. Użycie większej liczby atrybutów w jednym teście może prowadzić do uproszczenia drzewa. Należy zwrócić uwagę, iż proces doboru jest problemem znacznie trudniejszym i kosztowniejszym w realizacji. Złożoność obliczeniowa i skalowalność powstającego procesu klasyfikacji jest w tym przypadku priorytetem.

W poniższym tekście testy będziemy traktować jako funkcje zależne jedynie od atrybutu i jego wartości. Zachodzi konieczność wprowadzenia dodatkowych oznaczeń:

$A : X \to S_A$ - gdzie A atrybut:
$A(x)$ - wartość atrybutu A dla przykładu $x \in X$,
S_A - zbiór wartości atrybutu A,

[5] W praktyce najczęściej stosuje się *drzewa binarne*, w których każdy węzeł ma po dwóch potomków. Testy binarne to zatem testy o dwuelementowym zbiorze możliwych wyników.

$t : X \to S_t$ - gdzie t test:
$t(x)$ - wartość testu t dla przykładu $x \in X$,
S_t - zbiór wartości testu t.

Testy dla atrybutów nominalnych

Definicja 2.4.1 *Test $t : X \to S_t$ nazywamy testem tożsamościowym atrybutu $A : X \to S_A$ jeżeli:*

$$t(x) = A(x) \quad \forall\ x \in X \tag{2.8}$$

Jest to rodzaj testu, polegający na utożsamieniu testu z atrybutem. Oczywiście $S_t = S_A$. Taki test jest bardzo wygodny przy drzewach nie będących binarnymi. Pozwala na duży współczynnik rozgałęzienia, co zmniejsza głębokość drzewa i koszt klasyfikacji. Jego mankamentem jest niska stosowalność przy atrybutach o dużej liczbie możliwych wartości.

Definicja 2.4.2 *Test $t : X \to S_t$ nazywamy testem równościowym atrybutu $A : X \to S_A$ jeżeli:*

$$t(x) = \begin{cases} 0 & \textit{jeśli } A(x) = w \\ 1 & \textit{jeśli } A(x) \neq w \end{cases} \tag{2.9}$$

gdzie $w \in S_A$.

W tym przypadku $S_t = \{0, 1\}$. Wybór najlepszego testu równościowego wymaga sprawdzenia co najwyżej wszystkich wartości atrybutu A.

Definicja 2.4.3 *Test $t : X \to S_t$ nazywamy testem przynależnościowym atrybutu $A : X \to S_A$ jeżeli:*

$$t(x) = \begin{cases} 0 & \textit{jeśli } A(x) \in W \\ 1 & \textit{jeśli } A(x) \notin W \end{cases} \tag{2.10}$$

gdzie $W \subset S_A$.

Ten rodzaj testów jest uogólnieniem testów równościowych. Zauważmy, że dobór najlepszego testu wymaga co najwyżej sprawdzenia wszystkich właściwych podzbiorów zbioru S_A, co przy n możliwych wartościach atrybutu A wymaga $2^{n-1} - 1$ porównań. Jest to zależność wykładnicza (czyli bardzo kosztowna), sugerująca konieczność zaproponowania rozsądnego sposobu wyboru rozpatrywanych zbiorów W jako podzbiorów zbioru S_A. Przy tego rodzaju testach [6] jest to kwestia mająca kluczowy wpływ na dalszą skalowalność procesu klasyfikacji.

Testy dla atrybutów ciągłych

Przy atrybutach ciągłych można stosować testy przynależnościowe. W tym przypadku jako podzbiory $W \subset S_A$ bierze się pewne przedziały, gdzie dobór ich „końców" jest istotny. Mankamentem testów przynależnościowych przy ciągłych atrybutach, jest brak uwzględnienia istnienia relacji porządku w zbiorze możliwych wartości analizowanego atrybutu. Konstruuje się również testy uwzględniające istnienie owej relacji, nazywane testami nierównościowymi[7].

[6] Testy przynależnościowe stosowane są przy konstrukcji klasyfikatora SLIQ i SPRINT
[7] Testy nierównościowe są wykorzystywane przy konstrukcji klasyfikatora SLIQ i SPRINT

Definicja 2.4.4 *Test $t : X \to S_t$ nazywamy testem nierównościowym atrybutu $A : X \to S_A$ jeżeli:*

$$t(x) = \begin{cases} 0 & \text{jeśli } A(x) \leqslant w \\ 1 & \text{jeśli } A(x) > w \end{cases} \tag{2.11}$$

gdzie $w \in S_A$.

Zapisując $S_A = \{w_1, w_2, \ldots, w_n\}$ i przyjmując, że ciąg $\{w_1, w_2, \ldots, w_n\}$ jest ciągiem uporządkowanym (posortowanym w kolejności rosnącej), możemy stwierdzić, że dowolna taka wartość w, że $w_i < w < w_{i+1}$ dla ustalonego $i = 1, \ldots, n-1$, daje jednakowy wynik testu nierównościowego (dzieli zbiór X zawsze w taki sam sposób). Zatem, aby wybrać najbardziej odpowiedni test, wystarczy przeprowadzić tylko $n-1$ porównań. Zazwyczaj za punkt podziału obiera się środek przedziału $[w_i, w_{i+1}]$. Przy rozważaniu kwestii skalowalności, należy zwrócić uwagę na koszt sortowania zbioru wartości testowanego atrybutu[8].

2.4.2. Kryteria jakości podziałów

Podpróba docierająca do węzła dzielona jest na części. Oczywiście nie powinien to być proces przypadkowy. Zależy nam na podziale, który daje jak najmniejszą różnorodność klas w otrzymanych częściach, tak aby różnica pomiędzy różnorodnością klas w węźle i różnorodnością klas w tych częściach, była możliwie duża.

Definicja 2.4.5 *Każdą funkcję*

$$\phi : G \subset [0,1]^g \to \mathbb{R} \quad \text{gdzie} \quad (p_1, p_2, \ldots, p_g) \in G \Leftrightarrow \sum_{k=1}^{g} p_k = 1 \tag{2.12}$$

spełniającą następujące warunki:

1. *ϕ przyjmuje wartość maksymalną w punkcie $\left(\frac{1}{g}, \frac{1}{g}, \ldots, \frac{1}{g}\right) \in G$.*
2. *ϕ osiąga minimum jedynie w punktach*

$$(1,0,0,\ldots,0), (0,1,0,\ldots,0), \ldots, (0,0,0,\ldots,1) \in G$$

3. *$\phi(p_1, p_2, \ldots, p_g)$ jest symetryczna ze względu na $p_1, p_2, \ldots, p_g$.*

nazywamy funkcją różnorodności klas.

Definicja 2.4.6 *Jeżeli wierzchołek $m \in V$ jest następnikiem węzła $n \in N_r^T$ to $n \succ m-$następnikiem zbioru przykładów $U \subseteq \mathcal{X}$ i $n \succ m-$następnikiem zbioru obiektów $S \subseteq \mathcal{I}$ nazywamy:*

$$U_{n \succ m} := \left\{ x \in U \ : \ t_n(x) = m \right\} \tag{2.13}$$

$$S_{n \succ m} := \left\{ i \in S \ : \ t_n(x^i) = m \right\} \tag{2.14}$$

$n \succ m-$następniki reprezentują przykłady ze zbioru U i obiekty ze zbioru S klasyfikowane testem t_n do następnika m węzła n.

[8] Przy konstrukcji klasyfikatora SLIQ, stosuje się sortowanie wstępne (ang. pre-sorting)

Wniosek 2.4.1 *Zachodzą równości:*

$$\mathcal{X}_{n\succ m} = t_n^{-1}(m)$$
$$\mathcal{I}_{n\succ m} = \bigcup_{x\in\mathcal{X}_{n\succ m}} \mathcal{I}_x$$

Twierdzenie 2.4.1 *$n \succ m-$następnik mierzalnego zbioru obiektów $S \subseteq \mathcal{I}$ jest zbiorem mierzalnym.*

Dowód: Należy pokazać, że zbiór $S_{n\succ m} \in \mathfrak{B}$ dla dowolnych $n, m \in V$ gdzie $n \succ m$. Oznaczmy

$$\mathcal{X}_S := \Big\{x \in \mathcal{X} \ : \ \mathcal{I}_x \subseteq S\Big\}$$

Pisząc $\mathcal{I}_x$ rozważamy warstwę przykładu x (def. 1.3.7) . Zbiór $\mathcal{X}_S$ zawiera wszystkie przykłady, których warstwa jest podzbiorem zbioru S. Niech

$$Z := S \setminus \bigcup_{x\in\mathcal{X}_S} \mathcal{I}_x$$

Z założenia zbiór $S \in \mathfrak{B}$. Unia rodziny zbiorów mierzalnych jest zbiorem mierzalnym, zatem

$$\bigcup_{x\in\mathcal{X}_S} \mathcal{I}_x \in \mathfrak{B}$$

Na podstawie twierdzenia A.2.1 własność w_5 (różnica zbiorów mierzalnych jest zbiorem mierzalnym) stwierdzamy, że $Z \in \mathfrak{B}$.

$$S = Z \cup \bigcup_{x\in\mathcal{X}_S} \mathcal{I}_x$$

Możliwe są dwa przypadki:

1. $Z = \emptyset$ - rozważany zbiór S jest unią podrodziny rodziny warstw $\mathcal{I}^{\mathcal{X}}$,
2. istnieje dokładnie jeden przykład $z \in \mathcal{X}$, że $Z \subset \mathcal{I}_z$ - zbiór Z jest mierzalnym podzbiorem warstwy pewnego przykładu.

Z definicji następnika zbioru

$$S_{n\succ m} := \Big\{i \in S \ : \ t_n(x^i) = m\Big\}$$

wynika bezpośrednio, że istnieje $U \subseteq \mathcal{X}_S$, że

$$S_{n\succ m} = \bigcup_{x\in U} \mathcal{I}_x \quad \text{lub} \quad S_{n\succ m} = Z \cup \bigcup_{x\in U} \mathcal{I}_x$$

$\Rightarrow S_{n\succ m} \in \mathfrak{B}$

■

Definicja 2.4.7 *Jeżeli $S \subseteq \mathcal{I}$ jest takim mierzalnym zbiorem obiektów, że $P(S) > 0$, to miarę różnorodności klas w zbiorze S określamy wzorem:*

$$q(S) := \phi\Big(p(1|S), p(2|S), \ldots, p(g|S)\Big) \tag{2.15}$$

gdzie ϕ jest funkcją różnorodności klas, a $p(k|S)$ prawdopodobieństwem klasy k pod warunkiem, że zaszło zdarzenie S:

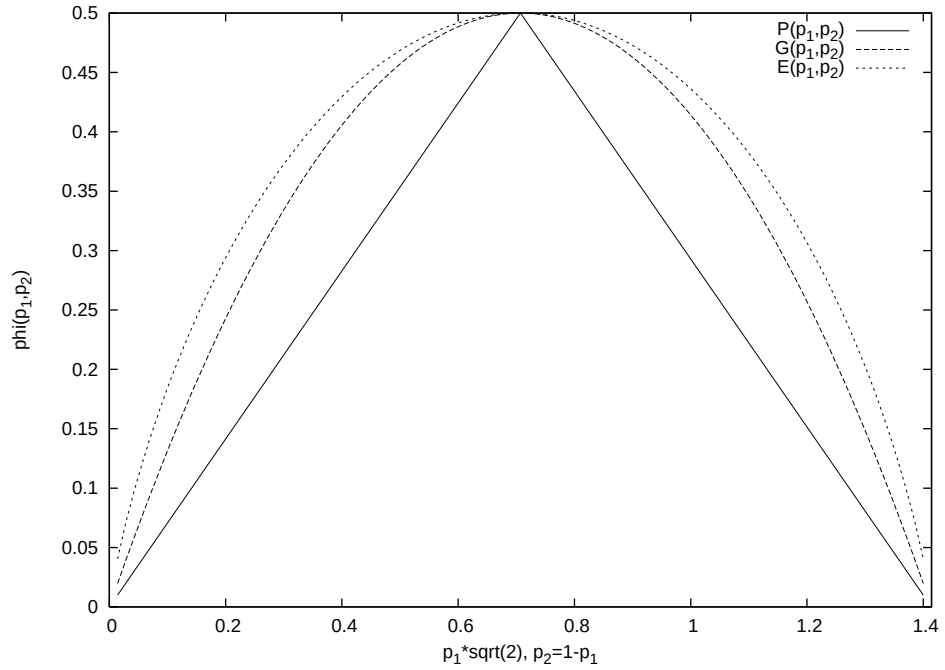
$$p(k|S) := P\Big(\mathcal{I}^k|S\Big) \tag{2.16}$$

Główne miary różnorodności klas

W praktyce najczęściej stosuje się niżej wymienione miary różnorodności klas. Indeks Giniego i entropie wykazują większą czułość na zmiany rozkładu klas w próbie.

1. Proporcja błędnych klasyfikacji:
$$q(S) \equiv p(S) := 1 - \max_k p(k|S) \tag{2.17}$$
2. Indeks Giniego:
$$q(S) \equiv G(S) := 1 - \sum_{k=1}^{g} \Big(p(k|S)\Big)^2 \tag{2.18}$$
3. Entropia:
$$q(S) \equiv E(S) := -\sum_{k=1}^{g} p(k|S) \ln p(k|S) \tag{2.19}$$

Rysunek 2.5 przedstawia zależność pomiędzy proporcją błędnych klasyfikacji, indeksem Giniego i entropią. Wartość funkcji entropii podzielona została przez $2 \ln 2$.



Rysunek 2.5: Proporcja błędnych klasyfikacji, indeks Giniego i entropia

Różnorodność jest tym większa im większa jest wartość miary $q(S)$. Po dokonaniu podziału w węźle $n \in N_r^T$ zbiór $S_{n\succ m}$ reprezentuje obiekty, które przeszły z węzła n do jego następnika $m \in n_\succ$.

Definicja 2.4.8 *Przez miarę zmiany różnorodności klas w węźle $n \in N_r^T$ drzewa klasyfikacyjnego T_r przy założeniu, że w węźle n znajdują się wszystkie obiekty z S, rozumie się kryterium oceny podziału w węźle n:*
$$\Delta q(S|n) := q(S) - \sum_{m \in n_\succ,\ P(S_{n\succ m})>0} P(S_{n\succ m}|S) q(S_{n\succ m}) \tag{2.20}$$

Pisząc $\Delta q(S|n)$ zakładamy istnienie testu w węźle n. W sytuacji, gdy do węzła przyporządkowany jest zbiór testów, definicja 2.4.8 umożliwia wybór podziału z największą wartością miary zmiany różnorodności klas. W tym sensie jest to podstawowe kryterium oceny testu w węźle drzewa klasyfikacyjnego.

Dla drzew binarnych Breiman [3] sformułował i udowodnił następujące twierdzenie.

Twierdzenie 2.4.2 (Breiman) *Dla binarnego drzewa T_r i wklęsłej funkcji różnorodności klas zachodzi:*

(i) $\Delta q(S|n) \geqslant 0$ *dla dowolnego węzła* $n \in N_r^T$ *oraz* $S \in \mathfrak{B}$*, że* $P(S) > 0$*,*

(ii) jeżeli $n_{\succ} = \{n_L, n_R\}$ *to równość w (i) zachodzi wtedy i tylko wtedy, gdy rozkłady klas w* S*,* $S_{n\succ n_L}$ *i* $S_{n\succ n_R}$ *są identyczne, tzn.:*

$$\forall\ k \in \{1, \ldots, g\} \qquad p(k|S) = p(k|S_{n\succ n_L}) = p(k|S_{n\succ n_R})$$

2.4.3. Kryterium stopu i reguła decyzyjna

Budowę drzewa klasyfikacyjnego rozpoczynamy od drzewa złożonego z jednego wierzchołka, do którego przyporządkowujemy zbiór uczący i zbiór dostępnych testów. W dalszych krokach konstruujemy podziały, tworząc węzły i ich następniki. Wraz ze wzrostem drzewa maleje zbiór uczący i zbiór testów docierający na kolejne jego poziomy. Poniżej przedstawiamy kilka oczywistych wytycznych, którymi należy się kierować podczas budowy drzewa. Należy zaniechać konstrukcji podziału w wierzchołku jeżeli:

1. Wystąpienie klasy k w podpróbie uczącej dostępnej w wierzchołku jest zdarzeniem z prawdopodobieństwem warunkowym 1.
2. Zastosowanie każdego dostępnego podziału daje zerową lub ujemną miarę zmiany różnorodności klas.
3. Zbiór dostępnych testów jest pusty.

Gdy obiekty w wierzchołku należą do tej samej klasy, to zajdzie przypadek 1. Sytuacja 2 ma miejsce w wierzchołku, w którym zbiór dostępnych testów jest oparty o atrybuty z jednakową wartością dla wszystkich dostępnych przykładów. Warunek 3 bezpośrednio wiąże się z brakiem uzasadnienia dla więcej niż jednokrotnego użycia danego podziału w obrębie jednej scieżki. Wystąpienie przypadków 2 lub 3 może świadczyć o zajściu jednej z poniższych sytuacji:

- zbiór trenujący nie jest poprawny i zawiera przekłamania,
- zestaw atrybutów nie opisuje obiektów w dostatecznym stopniu i w związku z tym przestrzeń hipotez jest zbyt uboga do reprezentowania pojęcia docelowego,
- przyjęty zbiór dostępnych atrybutów jest niewystarczający.

Definicja 2.4.9 *Jeżeli S jest podpróbą uczącą dostępną w wierzchołku n, a $\mathbb{T}$ zbiorem dostępnych testów, to kryterium stopu wstrzymujące konstrukcję podziału w n określamy wyrażeniem:*

$$\Big(\exists_{k\in C}\ p(k|S) = 1\Big) \vee \Big(\forall_{t_n\in\mathbb{T}}\ \Delta q(S|n) \leqslant 0\Big) \vee \Big(\mathbb{T} = \emptyset\Big) \tag{2.21}$$

Po wstrzymaniu konstrukcji podziału wierzchołek staje się liściem, do którego należy przyporządkować etykietę klasy.

Definicja 2.4.10 *Jeżeli S jest zbiorem uczącym, to etykietę daną wzorem*

$$c_S := \arg\max_k p(k|S) \tag{2.22}$$

nazywamy etykietą większościowej kategorii w zbiorze uczącym S.

W sytuacji, gdy zbiór dostępnych obiektów jest zdarzeniem z niezerowym prawdopodobieństwem etykietę wybieramy na podstawie etykiety większościowej kategorii.

Jeżeli nie istnieją wierzchołki, w których kryterium stopu uzna za zasadne utworzenie nowego podziału, to drzewo uznajemy za zbudowane.

2.4.4. Zstępująca konstrukcja drzewa

Istnieje wiele algorytmów uczenia się pojęć wykorzystujących drzewa decyzyjne do reprezentacji hipotez. Każdy z nich dąży do uzyskania struktury o niewielkim rozmiarze z możliwie niewielkim błędem próbki zakładając, że błąd rzeczywisty będzie również nieznaczny. Poniżej przedstawimy część wspólną większości tych algorytmów, określaną mianem *zstępującej konstrukcji drzewa*. Schemat zakłada rozpoczęcie budowy drzewa od pojedynczego wierzchołka (korzenia), któremu przyporządkowuje się wszystkie elementy ze zbioru trenującego. Kolejnym krokiem jest ustalenie zasadności utworzenia podziału w węźle. W przypadku decyzji pozytywnej kreujemy wierzchołki następniki. Nowoutworzone wierzchołki traktujemy jako korzenie „nowych" drzew z przypisanymi częściami próby uczącej. Procedurę powtarzamy do uzyskania liści (wierzchołków bez podziału), dla których określamy regułę decyzyjną. Tak rozumiany schemat najwygodniej jest opisać regułą rekurencyjną. Funkcja *buduj-drzewo* (alg.: 2.1) skonstruuje drzewo klasyfikacyjne T_r na podstawie zbioru uczącego $\mathbf{S}$ oraz zbioru dostępnych testów $\mathbf{T}$.

2.5. Problem nadmiernego dopasowania

Nadmierne dopasowanie do danych trenujących przejawia się bardzo małym błędem klasyfikacji (często nawet zerowym) na próbie uczącej, lecz zbyt dużym błędem rzeczywistym. Prawdopodobnie drzewo takie, poprzez bardzo złożoną strukturę, odzwierciedla przypadkowe zależności występujące w zbiorze uczącym. Z powodu swej struktury, umożliwiającej reprezentację dowolnej hipotezy, drzewa klasyfikacyjne są szczególnie narażone na ten problem. Błąd rzeczywisty nadmiernie dopasowanych drzew można zmniejszyć przez ich uproszczenie nazywane *przycięciem*. Drzewo przycięte ma prostszą strukturę, co daje krótszy czas klasyfikacji. Oczywiście dokładność klasyfikacji zbioru uczącego się pogarsza.

2.5.1. Schemat przycinania

W uproszczeniu proces przycinania polega na zastąpieniu drzewa wyjściowego jego poddrzewem. Formułując to bardziej obrazowo powiemy, że „ucina" się niektóre poddrzewa drzewa wyjściowego, zastępując je liśćmi, którym przypisuje się etykietę większościowej kategorii wśród obserwacji związanych z tym poddrzewem. Zamiast przycinać drzewo, możemy, poprzez modyfikację kryterium stopu, zapobiegać jego nadmiernemu wzrostowi. Takie postępowanie określa się *przycinaniem w trakcie wzrostu*. Znalezienie odpowiedniego kryterium stopu

Algorytm 2.1 Schemat zstępującego konstruowania drzewa

```
funkcja buduj-drzewo(n,S,c,𝕋)
   argumenty wejściowe:
     n - wierzchołek,
     S - zbiór trenujący,
     c - domyślna etykieta kategorii,
     𝕋 - zbiór możliwych testów;
   zwraca: drzewo decyzyjne reprezentujące hipotezę;

    1. jeśli kryterium-stopu(S,𝕋) to
    2.   c_n :=kategoria(S,c);
    3.   zwróć liść n;
    4. koniec jeśli
    5. t_n :=wybierz-test(S,𝕋);
    6. utwórz następniki n_≻;
    7. c_n :=kategoria(S,c);
    8. dla wszystkich m ∈ n_≻ wykonaj
    9.   buduj-drzewo(m,S_{n≻m},c_n,𝕋 \ t_n)
   10. koniec dla

wywołanie: buduj-drzewo(r,S,c_S,T)
```

okazuje się jednak trudne i częściej przycina się drzewa uprzednio zbudowane.

Przycinanie odbywa się z pomocą zbioru etykietowanych przykładów, zwanego *zbiorem przycinania*. Pełni on ważną funkcję przy szacowaniu błędu rzeczywistego przyciętego drzewa. Wyróżnia się dwa typy zbiorów przycinania:

1. Zbiór przycinania pochodzi spoza próby uczącej - jeśli mamy dostatecznie duży zbiór uczący.
2. Zbiór przycinania równy jest próbie uczącej - jeżeli nie dysponujemy dużym zbiorem uczącym.

Obecnie znanych jest wiele metod przycinania drzew. Algorytm 2.2 przedstawia funkcję opisującą część wspólną większości z nich.

2.5.2. Przycinanie MDL

Ideą zasady MDL[9] jest minimalizowanie sumy kosztu opisu danych przez model i kosztu opisu modelu. Jeżeli M jest modelem opisującym (kodującym) dane D, to całkowity koszt opisu definiujemy jako sumę

$$cost(M, D) := cost(D|M) + cost(M) \tag{2.23}$$

gdzie $cost(D|M)$ jest kosztem opisu danych D przez model M, a $cost(M)$ jest kosztem opisu modelu M. W kontekście drzew decyzyjnych modelami są drzewa otrzymane z przycinania drzewa wyjściowego. Opisywane (kodowane) dane to próba ucząca. Jako funkcję kosztu opisu

[9]MDL - Minimum Description Length.

Algorytm 2.2 Schemat przycinania drzewa klasyfikacyjnego

```
funkcja przytnij-drzewo(T_r, S_p)
   argumenty wejściowe:
     T_r - drzewo klasyfikacyjne,
     S_p - zbiór przycinania,
   zwraca: przycięte drzewo;

    1. dla wszystkich węzłów n ∈ N_r^T wykonaj
    2.    zastąp węzeł n liściem z etykietą większościowej kategorii
          jeśli nie powiększy to szacowanego na podstawie S_p
          błędu rzeczywistego drzewa T_r;
    3. koniec dla

wywołanie: przytnij-drzewo(T_r, S_p)
```

przyjmuje się najczęściej liczbę bitów wykorzystanych do jego reprezentacji.

Wcześniejsze implementacje zasady MDL w przycinaniu drzew pokazały, że wynikowe drzewa były nadmiernie przycięte (Quinlan i Rivest - 1989, Wallace i Patrick - 1993). Miało to oczywiście negatywne odbicie w jakości klasyfikacji opartej na takim drzewie. Jednak już w 1995 roku Mehta, Rissanen i Agrawal zaproponowali inną implementację przycinania drzew w oparciu o zasadę MDL. Metoda umożliwiła konstrukcję małych drzew bez znacznej utraty jakości klasyfikacji. Zastosowany algorytm miał pewne ograniczenia. Ponadto w danym węźle możliwe było jedynie „odcięcie" wszystkich potomków lub żadnego. Metoda przedstawiona poniżej jest wolna od tych ograniczeń.

Algorytm przycinania MDL dzieli się na dwie części: *schemat kodowania*, który wyznacza koszt (rozmiar) opisu danych oraz modelu, i *algorytm porównywania różnych drzew przyciętych*.

Koszt opisu danych przez model

Koszt opisu zbioru uczącego S przez drzewo T definiujemy jako sumę wszystkich błędnych klasyfikacji (klasa wyprodukowana przez drzewo jest inna od oryginalnej). Liczba błędnych klasyfikacji jest ustalana podczas fazy budowy drzewa. Nie jest więc to proces istotnie wpływający na złożoność algorytmów. Dalej liczbę błędnych klasyfikacji w węźle n oznaczamy przez $\mathrm{err}(n)$.

Koszt opisu modelu

Modelem jest drzewo zawierające podziały. Należy zatem wyznaczyć koszt opisu drzewa i koszty opisu testów zastosowanych w każdym węźle drzewa.

Węzeł drzewa może posiadać potomki w liczbie: jeden, dwa, lub zero (liść). Jako koszt opisu drzewa bierzemy sumaryczną liczbę bitów, która jest zależna od struktury drzewa. Wyróżnia się trzy podejścia do opisu struktury drzewa, gdzie przez $L(n)$ oznacza się koszt struktury węzła n. Dla węzła zezwala się na posiadanie:

- *typ 1*: $L(n) = 1$ (0 lub 2 potomków), 2 możliwości $\Rightarrow$ 1 bit niezbędny do opisu,
- *typ 2*: $L(n) = 2$ (0 lub 2 potomków, potomka lewego, potomka prawego), 4 możliwości $\Rightarrow$ 2 bity niezbędne do opisu),
- *typ 3*: $L(n) = \log 3$ (lewy potomek, prawy potomek, lub oba), tylko dla węzłów wewnętrznych, 3 możliwości $\Rightarrow \log 3$ bitów niezbędnych do opisu.

Koszt opisu podziałów uzależniony jest od typu testowanego atrybutu i wyznaczany jest osobno dla atrybutów ciągłych i nominalnych.

Jeżeli mamy do czynienia z testem nierównościowym postaci $A(x) \leqslant w$, gdzie A jest atrybutem numerycznym, a w liczbą rzeczywistą, to koszt testu definiujemy jako maksymalny koszt opisu liczby w. Koszt taki powinien być wyznaczony niezależnie dla każdego testu nierównościowego wykorzystanego w drzewie.

Dla testów przynależnościowych postaci $A(x) \in W$, gdzie W jest pewnym podzbiorem wartości nominalnego atrybutu A, koszt ustalamy w dwóch krokach. Najpierw wyznaczamy liczbę testów w drzewie opartych na danym atrybucie – powiedzmy n_A (tę czynność wykonujemy dla każdego atrybutu nominalnego). Koszt opisu testu definiujemy dalej jako

$$\ln n_A$$

Przez $L_{test}(n)$ oznaczamy koszt opisu testu w węźle n.

Algorytm przycinania MDL

Algorytm przycinania MDL wyznacza koszt (rozmiar, długość) opisu w każdym węźle drzewa klasyfikacyjnego. Wartość ta służy następnie do podjęcia decyzji czy mamy w danym węźle:

- utworzyć liść,
- przyciąć lewy, prawy potomek lub oba,
- pozostawić węzeł bez zmian.

Jeżeli $C(n)$ oznacza koszt opisu w węźle n, to kryterium przycinania przyjmuje postać:

1. $C_{leaf}(n) := L(n) + \mathrm{err}(n)$ - jeżeli n ma być liściem.
2. $C_{both}(n) := L(n) + L_{test}(n) + C(n_1) + C(n_2)$ - jeżeli n ma posiadać 2 potomki.
3. $C_{left}(n) := L(n) + L_{test}(n) + C(n_1) + \overleftarrow{C}(n_2)$ - jeżeli n ma posiadać tylko potomka lewego.
4. $C_{right}(n) := L(n) + L_{test}(n) + \overleftarrow{C}(n_1) + C(n_2)$ - jeżeli n ma posiadać jedynie potomka prawego.

$\overleftarrow{C}(n_1)$, $\overleftarrow{C}(n_2)$ oznaczają koszt opisu wyznaczany w przypadku częściowego przycięcia (lewy lub prawy potomek zostaje przycięty). W takim wypadku obserwacje przechodzące wcześniej przez przyciętą gałąź, są teraz klasyfikowane na podstawie statystyk z węzła macierzytego n. $\overleftarrow{C}(n_1)$, $\overleftarrow{C}(n_2)$ reprezentują koszt opisu obserwacji należących do węzłów potomków przy użyciu tych wcześniejszych statystyk.

Wyróżnia się strategie przycinania drzewa:

1. Strategia *pełna* - wykorzystująca *typ 1*, czyli 1 bit dla każdego węzła. Jeżeli $C_{leaf}(n) < C_{both}(n)$ (opcje 1 i 2) to węzeł zamieniamy na liść.
2. Strategia *częściowa* - wykorzystująca *typ 2*, czyli dwa bity dla każdego węzła. Węzeł jest konwertowany według tej opcji (spośród 1, 2, 3, 4), która ma najmniejszy koszt opisu.
3. Strategia *mieszana* - przewidująca dwie fazy. Faza pierwsza to strategia pełna. Faza druga używa jedynie opcji: 2, 3 oraz 4.

2.6. Zalety i ograniczenia drzew klasyfikacyjnych

Tak jak każda struktura reprezentacji, również drzewa klasyfikacyjne posiadają zalety i ograniczenia. W tym przypadku te pierwsze zdecydowanie przeważają nad drugimi.

Do zalet drzew klasyfikacyjnych zaliczamy:

1. *Możliwość reprezentacji dowolnej hipotezy* - konstrukcja drzewa klasyfikacyjnego umożliwia reprezentację dowolnego pojęcia, którego definicję można wyrazić w zależności od atrybutów użytych do opisu przykładów.
2. *Efektywność procesu klasyfikacji* - stosowanie hipotezy uzyskanej w wyniku uczenia się jest bardzo efektywne. W typowych przypadkach czas jest liniowo ograniczony przez liczbę obserwacji i/lub liczbę atrybutów.
3. *Efektywność dla dużych zbiorów uczących* - aktualnie drzewa klasyfikacyjne wykorzystuje się najczęściej i najchętniej do rozwiązywania problemów z ogromnymi ilościami danych.
4. *Czytelność reprezentacji* - o ile drzewa nie są zbyt duże i złożone.

Jako ograniczenia drzew klasyfikacyjnych wymienić można:

1. *Ryzyko dużej złożoności drzewa* - dla złożonych hipotez drzewa mogą być również bardzo złożone. Wiąże się to z tym, że zazwyczaj testy stosowane do budowy drzewa, wykorzystują tylko jeden atrybut, co implikuje utratę zależności pomiędzy atrybutami.
2. *Reprezentacja alternatyw warunków* - drzewo klasyfikacyjne, przez swą budowę, naturalnie narzuca stosowanie koniunkcji warunków.
3. *Brak łatwej możliwości inkrementacyjnego aktualizowania* - po dodaniu nowych przykładów trudno jest zaktualizować już istniejące drzewo. Istnieją oczywiście algorytmu realizujące to zadanie, jednak są dość kosztowne, a zaktualizowane drzewo ma zazwyczaj mniejszą precyzję, niż drzewo zbudowane na nowo.

Rozdział 3

Klasyfikator SLIQ

3.1. Wprowadzenie

Większość algorytmów klasyfikujących powstała w dość odległej przeszłości, gdy jeszcze nie istniały ogromne zbiory danych trenujących. Proces uczenia projektowano bez szczególnego nacisku na czynniki wydajnościowe. Poniżej przedstawiamy metodę klasyfikacji SLIQ[1] [4], wykorzystującą drzewa decyzyjne do reprezentacji hipotez. SLIQ umożliwia użycie danych opisanych atrybutami numerycznymi oraz nominalnymi. Zastosowana w nim nowatorska technika sortowania wstępnego, w połączeniu z poziomą strategią wzrostu drzewa, czyni SLIQ niezwykle efektywnym i skalowalnym narzędziem dla ogromnych prób trenujących, danych dyskowych, nie mieszczących się w pamięci operacyjnej, z dużą liczbą atrybutów i klas. SLIQ ponadto wykorzystuje przycinanie MDL. Autorami SLIQ są: Manish Mehta, Rakesh Agrawal oraz Jorma Rissanen (1996).

3.2. Struktury danych

Właściwości SLIQ w dużej mierze są konsekwencją specjalnej konstrukcji struktur danych. Listy wartości atrybutów umożliwiają ograniczenie się do jednokrotnego sortowania każdego z nich. Lista klas pozwala w łatwy sposób dotrzeć do odpowiedniego węzła budowanego drzewa. Histogramy efektywnie wyznaczające miary różnorodności klas. Wszystko to składa się na dużą skalowalność całego procesu uczenia.

Lista klas

Lista klas jest to zbiór par <*etykieta-klasy, referencja-do-liścia*>. Elementów na liście klas jest dokładnie tyle ile jest obserwacji w próbie uczącej. Zakłada się, że $i - ty$ element listy klas odpowiada $i - tej$ obserwacji z próby uczącej (zachodzi zgodność w etykietach klas). Pole <*referencja-do-liścia*> zawiera odnośnik do jednego z liści, który klasyfikuje obserwacje do tej klasy. Listę klas ustawiamy początkowo dla wszystkich pozycji tak, aby wskazywała na korzeń. Lista klas powinna pomieścić się w pamięci operacyjnej. Rysunek 3.1 przedstawia przykładową strukturę listy klas.

Lista wartości atrybutu

[1] SLIQ - Supervised Learning in Quest (Quest - projekt badawczy IBM).

Próba ucząca (dla atrybutu A)

Num. obs.	Wartość atrybutu	Etykieta klasy
1		
2		
...		
n		

Lista wartości (dla atrybutu A)

Num. obs.	Wartość atrybutu	Referencja do listy klas
1		
2		
...		
n		

Lista klas (dla próby uczącej)

Num. obs.	Etykieta klasy	Referencja do liścia
1		
2		
...		
n		

Rysunek 3.1: Struktura danych

Lista wartości atrybutu jest to zbiór par <*wartości-atrybutu, referencja-do-listy-klas*>. Każda wartość atrybutu na liście wartości jest jednoznacznie powiązana z jedną obserwacją z próby uczącej, w konsekwencji również z etykietą klasy. Dla danej wartości atrybutu odnajdujemy odpowiadającą jej etykietę klasy, co umożliwia przejście do listy klas. Listy wartości tworzone są niezależnie dla każdego atrybutu numerycznego i mogą być zapisywane na dysku. Rysunek 3.1 przedstawia przykładową strukturę listy wartości atrybutu.

Histogramy

Histogramy wykorzystywane są do określenia rozkładu częstości (proporcji) klas w węzłach. Histogram dla atrybutu numerycznego ma inną strukturę od histogramu dla atrybutu nominalnego. Do określenia jakości podziału nierównościowego wystarczy informacja ile obserwacji z danej klasy spełniło test, a ile nie. Przy wykorzystaniu testu przynależnościowego pojawia się kwestia wyboru podzbioru. W tym celu histogram przechowuje rozkład częstości klas w węzłach potomkach w podziale na wartości atrybutu. Rysunek 3.2 prezentuje strukturę histogramu dla atrybutu numerycznego i nominalnego.

Histogram (atrybuty ciągłe / numeryczne)

	Klasa 1	Klasa 2	...	Klasa n
L	[częstość]	[częstość]	...	[częstość]
R	[częstość]	[częstość]	...	[częstość]

Histogram (atrybuty nominalne)

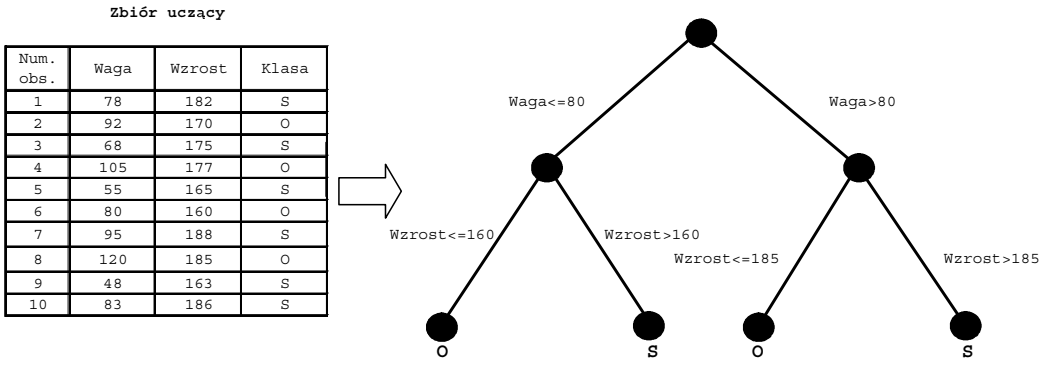
		Klasa 1	Klasa 2	...	Klasa n
Wartość 1	L	[częstość]	[częstość]	...	[częstość]
	R	[częstość]	[częstość]	...	[częstość]
Wartość 2	L	[częstość]	[częstość]	...	[częstość]
	R	[częstość]	[częstość]	...	[częstość]
...	L	[częstość]	[częstość]	...	[częstość]
	R	[częstość]	[częstość]	...	[częstość]
Wartość k	L	[częstość]	[częstość]	...	[częstość]
	R	[częstość]	[częstość]	...	[częstość]

Rysunek 3.2: Struktura histogramów

Oznaczenia L i R określają statystyki dla obserwacji, które spełniają kryterium podziału oraz dla tych, które kryterium tego nie spełniają. Niezależne histogramy przyporządkowane są do każdego liścia aktualnie budowanego drzewa.

3.3. Sortowanie wstępne

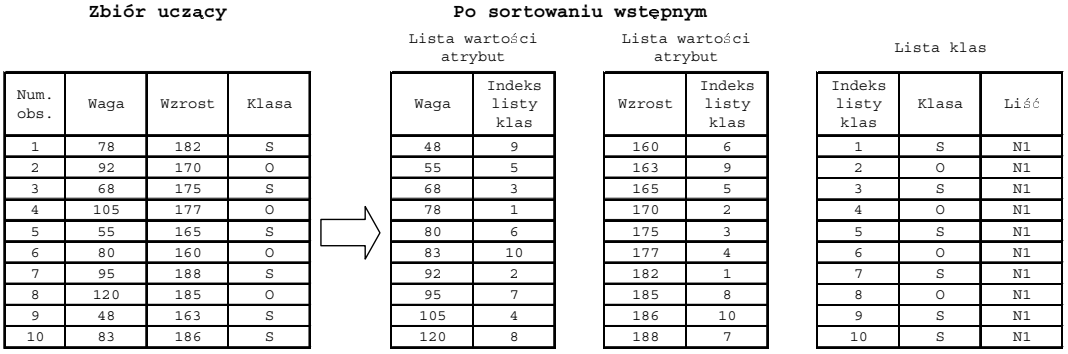
Rozważmy drzewo decyzyjne przedstawione na rysunku 3.3 - dwa atrybuty (Waga, Wzrost), 10 obserwacji w próbie uczącej oraz 2 klasy (O,S).



Rysunek 3.3: Zbiór uczący i drzewo klasyfikacyjne

Do utworzenia drzewa decyzyjnego, potrzebujemy wyznaczenia wielu najlepszych podziałów (względem zadanych testów). Dla atrybutów ciągłych wiąże się to najczęściej z potrzebą sortowania danych, które dotarły do danego węzła. Takie podejście ma jednak bardzo negatywne konsekwencje ze względu na otrzymywaną skalowalność procesu klasyfikacji. Pierwsza technika zastosowana w klasyfikacji metodą SLIQ, pozwala rozwiązać problem. Nazywamy ją *sortowaniem wstępnym* (ang. *pre-sorting*). Schemat sortowania wstępnego zakłada konieczność sortowania danych tylko raz dla każdego atrybutu (sortowanie odbywa się na poziomie korzenia).

Rysunek 3.4 przedstawia powstałe dwie wstępnie posortowane listy wartości oraz wspólną listę klas. Zwróćmy uwagę, że referencje do liścia są ustawione na N1 - czyli korzeń.



Rysunek 3.4: Dane przed i po sortowaniu wstępnym

3.4. Pozioma strategia wzrostu

Metoda SLIQ wykorzystuje *poziomą strategią wzrostu* drzewa (ang. *breadth-first*). Wcześniej omówiliśmy techniki zstępujące (ang. *depth-first*), rekurencyjnie budujące drzewo, kierując się początkowo ku „dołowi". SLIQ w pierwszej kolejności tworzy cały poziom, poruszając się niejako w szerz, wyznaczając podziały dla wszystkich liści bieżącego drzewa.

Lista wartości każdego atrybutu przeszukiwana jest jednokrotnie. Dla atrybutów ciągłych in-

Algorytm 3.1 Wyznaczenie podziałów SLIQ

```
funkcja SLIQ-wyznacz-podziały
   zwraca: najlepsze podziały na danym poziomie drzewa;

 1. dla każdego atrybutu A wykonaj
 2.   rozważ listę wartości atrybutu A;
 3.   dla każdej wartości v z listy wartości atrybutu A wykonaj
 4.     znajdź odpowiadającą pozycję na liście klas;
 5.     przejdź do odpowiadającej klasy i dalej do liścia (powiedzmy l);
 6.     zaktualizuj histogram w liściu l;
 7.     jeśli A jest atrybutem ciągłym to
 8.       wyznacz indeks Giniego dla testu (A ⩽ v) w liściu l;
 9.     koniec jeśli
10.     jeśli A jest atrybutem nominalnym to
11.       dla każdego liścia drzewa
12.         znajdź podzbiór zbioru wartości A z najlepszym podziałem
12.         (w oparciu o Indeks Giniego);
13.       koniec dla
14.     koniec jeśli
15.   koniec dla
16. koniec dla
```

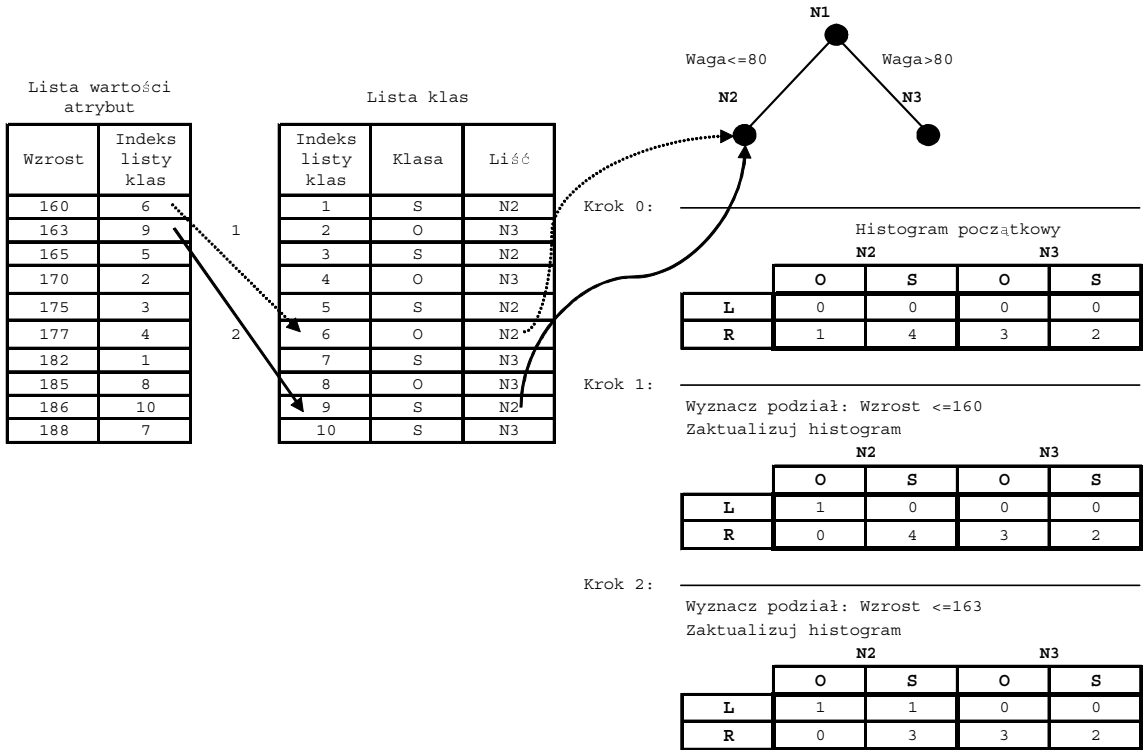
deks Giniego wyznacza się na bieżąco. Przy atrybutach nominalnych dopiero po zakończeniu skanowania listy jego wartości. Następnie wybierany jest podzbiór zbioru wartości atrybutu, dający najlepszy podział. Do wyznaczenia najlepszego podziału we wszystkich liściach wystarczy przejść jednokrotnie każdą listę wartości atrybutu.

Do podziałów opartych na atrybutach nominalnych, SLIQ wykorzystuje testy przynależnościowe. Wiemy już, że wyznaczenie wszystkich podzbiorów zbioru wartości atrybutu nominalnego, może być bardzo kosztowne. Do rozwiązania tego problemu użyto w SLIQ podejścia mieszanego. Dla małych zbiorów wartości (o małej liczności, mniejszej niż ustalony próg) wyznacza się wszystkie niezbędne podzbiory. Za ograniczenie górne liczności takiego zbioru przyjmuje się liczbę 10 (2^{10} - podzbiorów można wyznaczyć w miarę szybko). W przypadku dużych zbiorów stosuje się algorytm zachłanny. Algorytm zachłanny rozpoczyna od pustego podzbioru. W kolejnych krokach do tego podzbioru dodawane są elementy, które dają najlepszy podział. Proces ten trwa aż do momentu, kiedy nie obserwujemy już poprawy otrzymanych podziałów.

Rysunek 3.5 przedstawia metodę przejścia z listy wartości, poprzez listę klas, do liścia. Istotną kwestią jest aktualizacja statystyk w histogramach.

Utworzenie potomków i aktualizacja listy klas

Po wybraniu testu SLIQ tworzy potomki rozważanego węzła. Kolejnym krokiem jest aktualizacja listy klas tak, aby odzwierciedlała bieżącą status drzewa. Rysunek 3.6 prezentuje przykład realizacji wyżej opisanego zadania. Podział oraz następująca po nim aktualizacja listy klas, powtarzają się do momentu, gdy do każdego liścia w powstałym drzewie docierają

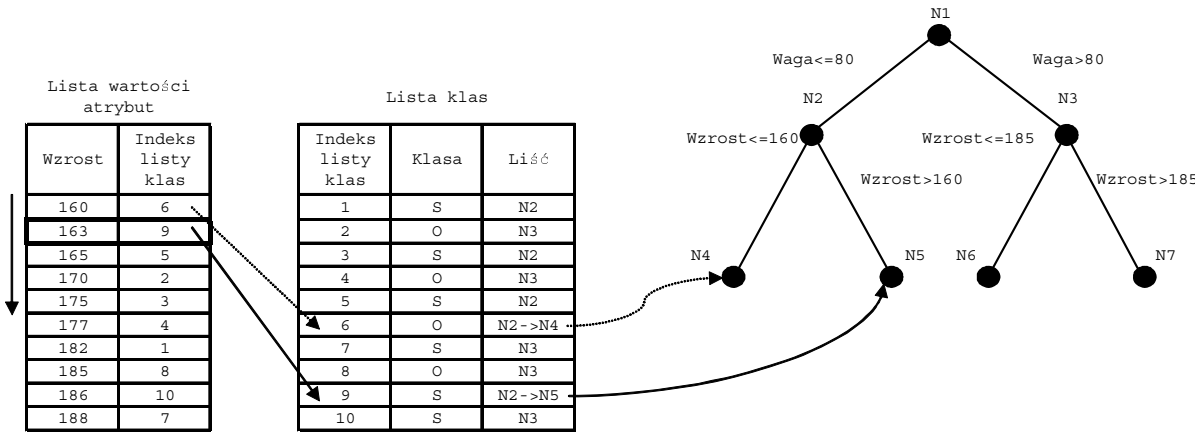


Rysunek 3.5: Przebieg algorytmu SLIQ

obserwacje tylko z jednej klasy. Należy zwrócić uwagę na fakt, że niektóre liście mogą osiągnąć ten stan wcześniej. W takim przypadku nie ma więc sensu stosowania dla nich kolejnych podziałów.

Algorytm 3.2 Aktualizacja listy klas

```
funkcja SLIQ-zaktualizuj-listę-klas
   zwraca: zaktualizowaną listę klas;

    1. dla każdego atrybutu A użytego w podziale wykonaj
    2.   rozważ listę wartości atrybutu A;
    3.   dla każdej wartości v z listy wartości atrybutu A wykonaj
    4.     znajdź odpowiadającą pozycję na liście klas (powiedzmy e);
    5.     znajdź nową klasę c dla v po zastosowaniu testu w węźle e;
    6.     zaktualizuj etykietę klasy z e na c;
    7.     zaktualizuj referencję do liścia w e na liść odpowiadający c;
    8.   koniec dla
    9. koniec dla
```



Rysunek 3.6: Aktualizacja listy klas

Rozdział 4

Klasyfikator SPRINT

4.1. Wprowadzenie

Przedstawiony w rozdziale 3 algorytm SLIQ zachowuje własność skalowalności przy założeniu, że lista klas rozważanego zbioru trenującego mieści się w pamięci operacyjnej komputera. Przypomnijmy, że rozmiar listy klas zależy jedynie od liczby przykładów w zbiorze uczącym. Tak skonstruowany element globalny, posiadający ograniczenie na rozmiar, jest zarazem największym mankamentem algorytmu. Poniżej omówimy klasyfikator o nazwie SPRINT[5][1] (J.C. Shafer, R. Agrawal, M. Mehta, 1996), uwolniony od tego ograniczenia, w dużej mierze podobny do metody SLIQ. SPRINT wykorzystuje bardzo podobne struktury danych do tych zaprojektowanych dla SLIQ. SPRINT nie definiuje listy klas. W konsekwencji otrzymano brak elementów globalnych, umożliwiając łatwe i efektywne zrównoleglenie implementacji. Przez pojęcie równoległej implementacji rozumiemy jednoczesne działanie algorytmu na wielu procesorach (CPU). Każdy procesor pracuje nad częścią danych w niezależny od reszty sposób. Celem jest uzyskanie jednego spójnego modelu.

4.2. Metoda podstawowa

Algorytm SPRINT działa w dwóch fazach: wzrostu drzewa binarnego oraz jego przycinania. Faza wzrostu jest tym, co różni SPRINT od metody SLIQ. Drzewo SPRINT przycinane jest metodą opartą o zasadę MDL, która została omówiona w sekcji 2.5.2 niniejszej pracy.

SPRINT wykorzystuje technikę depth-first zstępującej konstrukcji drzewa (2.4.4). Przypomnijmy, że drzewo SLIQ budowane jest w oparciu o poziomą strategię wzrostu (breath-first). Jako, że schemat zstępującej konstrukcji drzewa został bardzo szczegółowo opisany algorytmem 2.4.4, skoncentrujemy się jedynie na strukturach danych, pozwalających w łatwy sposób wybrać najlepszy podział.

Lista wartości atrybutu

SPRINT, podobnie jak SLIQ, dla każdego atrybutu tworzy listę jego wartości. Jest to zbiór trójek postaci: <*wartość-atrybutu, etykieta-klasy, referencja-do-przykładu*>. Początkowa każda lista wartości atrybutu, powiązana z korzeniem drzewa, zawiera dokładnie taką sama liczbę elementów co zbiór trenujący. Kolejnym krokiem jest sortowanie wstępne list atrybutów numerycznych, analogiczne do użytego w SLIQ (nie ma konieczności ponownego sortowania list

[1] Scalable Parallelizable Induction of Decision Trees

w trakcie fazy wzrostu drzewa). Listy atrybutów, w przypadku znacznych rozmiarów, mogą być przetrzymywane na dysku komputera. W SLIQ lista wartości atrybutu jest elementem globalnym algorytmu. SPRINT do każdego wierzchołka przyporządkowuje nową listę wartości atrybutu, otrzymywaną z podziału już istniejącej w węźle macierzystym. Proces podziału dokonuje się w trakcie zstępującego wzrostu poczynając od korzenia. Przykładowe listy wartości atrybutów po sortowaniu wstępnym zaprezentowane sa na rysunku 4.1. Rysunek 4.2 przedstawia podział list wraz z dokonaniem podziału przykładów węźle.

Zbiór uczący

Num. obs.	Waga	Wzrost	Klasa
1	78	182	S
2	92	170	O
3	68	175	S
4	105	177	O
5	55	165	S
6	80	160	O
7	95	188	S
8	120	185	O
9	48	163	S
10	83	186	S

Po sortowaniu wstępnym

Lista wartości atrybutu

Waga	Klasa	Num. obs.
48	S	9
55	S	5
68	S	3
78	S	1
80	O	6
83	S	10
92	O	2
95	S	7
105	O	4
120	O	8

Lista wartości atrybutu

Wzrost	Klasa	Num. obs.
160	O	6
163	S	9
165	S	5
170	O	2
175	S	3
177	O	4
182	S	1
185	O	8
186	S	10
188	S	7

Rysunek 4.1: Listy wartości atrybutów po sortowaniu wstępnym w algorytmie SPRINT

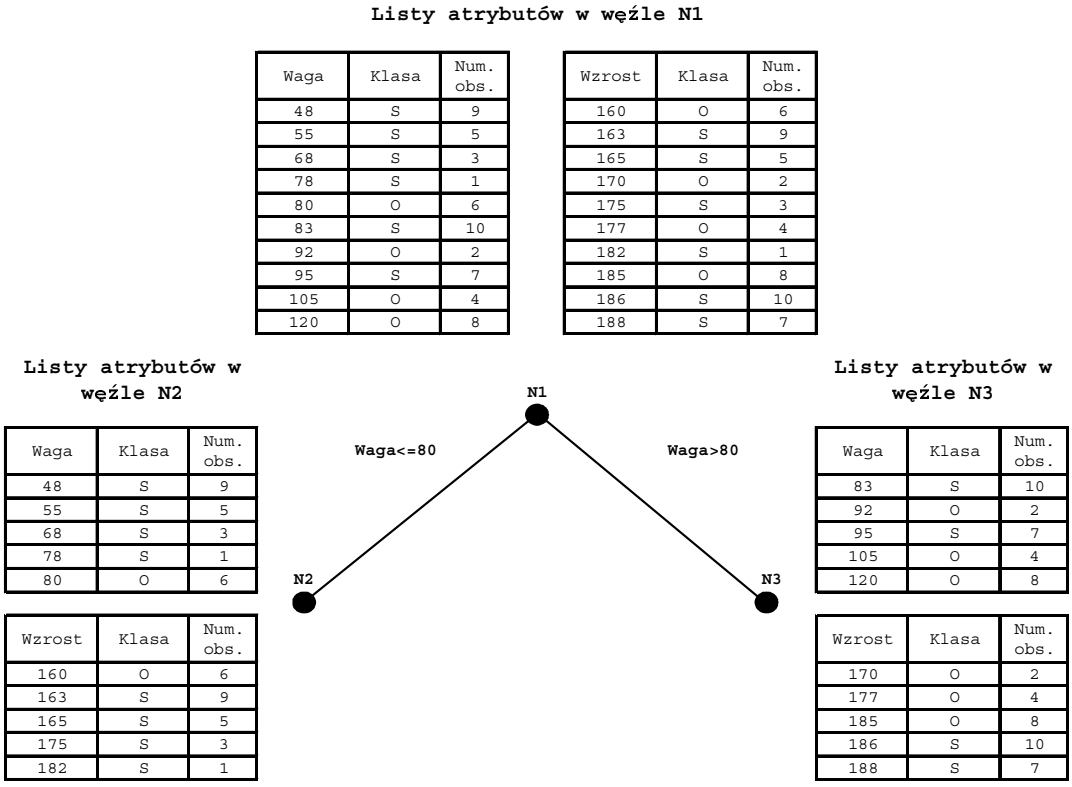
Histogramy

Kryterium oceny jakości podziałów wykorzystane w SPRINT oparte jest na indeksie Giniego (2.18). Jest to kolejne podobieństwo do metody SLIQ. Również struktury danych umożliwiające efektywne określenie wartości miary Giniego nie uległy zmianie w stosunku do SLIQ. SPRINT używa identycznych histogramów jak te opisane w sekcji 3.2.

Wybór najlepszego podziału

SPRINT, w odróżnieniu od SLIQ, dokonuje aktualizacji histogramów dla każdego węzła z osobna (w SLIQ, podczas jednokrotnego przejścia przez listy atrybutów, dochodzi do aktualizacji histogramów w każdym węźle aktualnie procesowanego poziomu drzewa). W przypadku testowania atrybutu numerycznego, do określenia najlepszego punktu podziału, wystarczy jednokrotnie rozważyć dostępną listę jego wartości. Dla atrybutów nominalnych SPRINT stosuje procedurę hybrydową (mieszaną). Przypomnijmy, że obie metody konstrukcję podziałów opierają na: testach nierównościowych (def. 2.4.4) w przypadku atrybutów numerycznych oraz testach przynależnościowych (def. 2.4.3) gdy rozważane są atrybuty nominalne.

Zastosowanie najlepszego z możliwych testów polega na utworzeniu potomków, oraz odpowiednim podziale dostępnych przykładów i list wartości atrybutów. Do efektywnego podziału list wykorzystuje się tablicę haszującą utworzoną na podstawie <*referencji-do-przykładu*> listy testowanego atrybutu. Rysunek 4.3 prezentuje tablicę haszująca utworzoną na podstawie podziału $Waga \leqslant 80$. W przypadku, gdy tablica haszująca nie mieści się w pamięci operacyjnej komputera, podział dokonywany jest w więcej niż jednym kroku.



Rysunek 4.2: Podział list wartości atrybutów podczas fazy wzrostu drzewa SPRINT

4.3. Metoda zrównoleglona

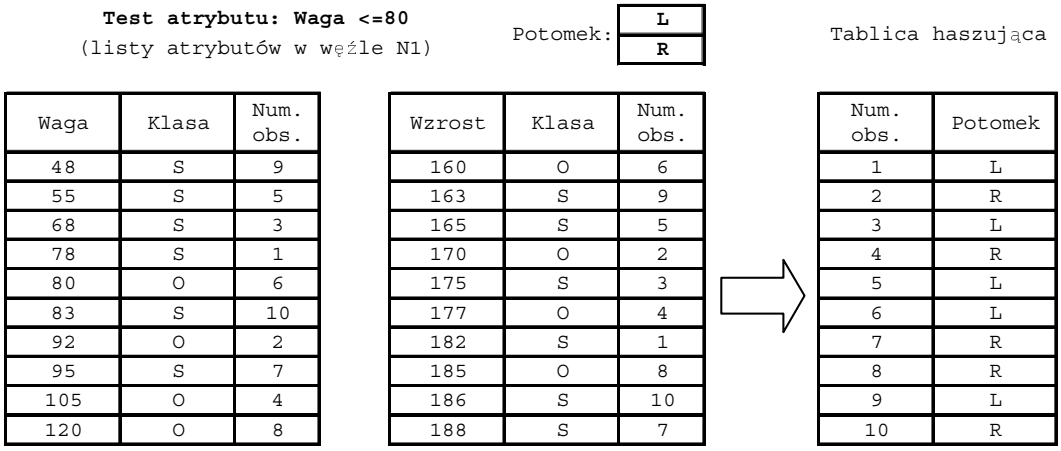
Klasyfikator SPRINT w wersji zrównoleglonej jest zaprojektowany tak, aby proces uczenia mógł przebiegać efektywnie przy jednoczesnym wykorzystaniu wielu jednostek obliczeniowych. Skupimy się jedynie na fazie wzrostu drzewa, gdyż przycinanie drzewa metodą MDL jest zadaniem obliczeniowo mało kosztownym i może być wykonane z użyciem jednego procesora.

Głównym założeniem jakiejkolwiek równoległej implementacji jest posiadanie środowiska, gdzie każdy z N procesorów posiada prywatną pamięć operacyjną oraz prywatną pamięć zaalokowaną na dyskach. Procesory połączone są w sieć komunikacyjną i mogą komunikować się jedynie przez wysyłanie komunikatów.

Analogicznie do podstawowej metody w zrównoleglonej fazie budowy drzewa podstawowym problemem jest konstrukcja i przeprowadzenie najlepszego podziału. Równie istotną kwestią jest rozmieszczenie (podział) danych oraz obliczeń.

Podział danych oraz obliczeń

Główne struktury danych w klasyfikatorze SPRINT to: lista wartości atrybutu i histogramy tworzone na podstawie próby uczącej. Efektywny podział danych oraz obliczeń osiągany jest przez podział próby uczącej na N możliwie równolicznych części. W kolejnym kroku każdy z N procesorów otrzymuje porcję danych, na podstawie której wyznacza własne listy wartości



Waga	Klasa	Num. obs.
48	S	9
55	S	5
68	S	3
78	S	1
80	O	6
83	S	10
92	O	2
95	S	7
105	O	4
120	O	8

Wzrost	Klasa	Num. obs.
160	O	6
163	S	9
165	S	5
170	O	2
175	S	3
177	O	4
182	S	1
185	O	8
186	S	10
188	S	7

Num. obs.	Potomek
1	L
2	R
3	L
4	R
5	L
6	L
7	R
8	R
9	L
10	R

Rysunek 4.3: Przykład tablicy haszującej

atrybutu. Następnie listy wartości dla atrybutów numerycznych są sortowane i dzielone na nowo tak, aby każdy z procesorów pracował nad „przylegającymi" do siebie wartościami. Taka procedura ma na celu uniknięcie sytuacji, gdzie jeden z procesów pracuje przykładowo nad wartościami $\{1, 5, 10\}$, a drugi przeprowadza obliczenia dla $\{3, 7, 12\}$. Po prawidłowym przeprowadzeniu sortowania wraz z podziałem w wyniku powinniśmy otrzymać: $\{1, 3, 5\}$ oraz $\{7, 10, 12\}$.

Wybór najlepszego podziału

Wybór najlepszego podziału w zrównoleglonej wersji SPRINT jest przeprowadzony w sposób bardzo podobny do metody podstawowej. Każdy z procesorów pracuję nad własną porcją danych prowadząc niezależne od innych obliczenia na przypisanych do niego strukturach danych. Dokonanie podziału wymaga dodatkowej komunikacji (wymiany danych) pomiędzy wszystkimi procesorami.

Nie ma konieczności wykonania żadnych dodatkowych zadań. SPRINT w swych założeniach został projektowany do przeprowadzania zrónoleglonych obliczeń.

4.4. Porównanie klasyfikatorów SPRINT i SLIQ

Algorytmy SPRINT i SLIQ budują identyczne drzewa decyzyjne, prowadząc do jednakowych hipotez. Jest to konsekwencją zastosowania tego samego kryterium oceny jakości podziału oraz identycznej metody przycinania drzewa binarnego. Na uwagę zasługuje różnica w strategii wzrostu drzewa, SLIQ wykorzystuje poziomą, SPRINT stosuje technikę zstępującą. SPRINT nie posiada elementów globalnych, stając się odpornym na rozmiar zbioru trenującego oraz umożliwiając łatwe zrównoleglenie działania. Tym samym SPRINT rozwiązuje dwa największe mankamenty metody SLIQ, tworząc szybkie, skalowalne i bardzo dokładne narzędzie służące eksploracji danych.

Rozdział 5

Implementacja klasyfikatora SLIQ

5.1. Wprowadzenie

Implementacja klasyfikatora SLIQ została przeprowadzona we współpracy z Instytutem Podstaw Informatyki Polskiej Akademii Nauk[1] w ramach rozwoju pakietu „`dmLab`". Jest to implementacja wykonana przez autora niniejszej pracy w oparciu o zawarty we wspomnianym pakiecie schemat reprezentacji danych trenujących / testowych. Celem implementacji było wykazanie użyteczności klasyfikatora SLIQ w rozwiązywaniu problemów, gdzie złożoność obliczeniowa i skalowalność procesu uczenia jest szczególnie istotna.

5.2. Środowisko uruchomieniowe

Algorytm SLIQ (analogicznie do całości pakietu `dmLab`) został zaimplementowany w języku JAVA[2]. Do pakietu `dmLab` dodano ponad 2000 linii kodu, który zawiera:

1. `dmLab.classifier` - abstrakcyjną implementację klasyfikatora,
2. `dmLab.classifier.tree` - abstrakcyjną implementację reprezentacji drzewa,
3. `dmLab.classifier.tree.treePruning` - abstrakcyjną implementację metody przycinania drzewa,
4. `dmLab.classifier.tree.sliqClassifier` - implementację algorytmu SLIQ,
5. `dmLab.classifier.tree.treePruning.mdlTreePruning` - implementację przycinania drzewa metodą MDL.

Uruchomienie pakietu wymaga zainstalowania środowiska *Java Runtime Environment*. Wykorzystanie języka JAVA zapewnia niezależność implementacji od architektury oraz platformy / systemu operacyjnego.

5.3. Analiza wyników

Dane źródłowe wykorzystane w testach niemal w całości utworzono sztucznie, wykorzystując prosty generator. Jedynie w przykładzie 5.3.1 dane zawierają prawdziwe relacje, względem poniższej definicji z atrybutem „play" jako zmienną grupującą:

[1] http://www.ipipan.waw.pl
[2] http://www.java.sun.com

```
#@relation weather
attributes
{
 outlook nominal
 temperature numeric
 humidity numeric
 windy nominal
 play nominal decision(all)
}
events
{
 ...
}
```

Przykład 5.3.1 prezentuje wynik działania klasyfikatora SLIQ z pakietu `dmLab`. W pierwszej części widzimy *macierz błędu* wraz z precyzją nieprzyciętego drzewa SLIQ. Następnie (sekcja „`Tree info`") odnajdujemy informacje o liczbie węzłów i liści w drzewie, jego wysokości, oraz czasach budowy i/lub przycinania. W kolejnym kroku system informuje o przeprowadzonym przycinaniu. Sekcja „`Tree structure`" prezentuje strukturę przyciętego drzewa SLIQ. Sekcja „`Tree info`" aktualizuje liczbowe wskaźniki dla przyciętego drzewa. Na końcu odnajdujemy jakość ostatecznej klasyfikacji.

Przykład 5.3.1 *Drzewo SLIQ zbudowane na próbie uczącej o liczności 2029.*

```
attributes: 5 events: 2029

Confusion Matrix
                System Answer
        no      yes     other
no      723     0       0
yes     1       1305    0
other   0       0       0

Accuracy : 0.9995071
------------- Tree info -------------
Nodes number ..... : 48
Leafs number ..... : 49
Tree height ..... : 10
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 0.071
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.071
================================================
MDL Pruning... Done!
================= Tree structure =================
Root:
outlook \in {overcast}
|  L:   [leaf] -> play = yes
|  R:   humidity <= 84.0
|  |  L:   temperature <= 65.0
|  |  |  L:   [leaf] -> play = no
|  |  |  R:   [leaf] -> play = yes
|  |  R:   temperature <= 70.0
|  |  |  L:   [leaf] -> play = yes
|  |  |  R:   [leaf] -> play = no
------------- Tree info -------------
Nodes number ..... : 4
Leafs number ..... : 5
Tree height ..... : 3
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 0.071
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.071
```

```
================================================
Testing... Done!
Confusion Matrix
                System Answer
        no      yes     other
no      1412    34      0
yes     10      2602    0
other   0       0       0

Accuracy : 0.9891572
```

Podsumowując przykład 5.3.1 powiemy, że przycinając drzewo drastycznie zmniejszyliśmy jego złożoność (liczba węzłów i liści) nie tracąc jakości klasyfikacji (spadek o 1%). Wynik ten bardzo dobrze obrazuje istotę przycinania drzew.

5.3.1. Skalowalność SLIQ

Skalowalność klasyfikacji metodą SLIQ zbadaliśmy poprzez analizę wpływu wzrostu liczności próby uczącej i wzrostu liczby atrybutów na czas uczenia. Tabela 5.1 prezentuje podsumowanie eksperymentu dla wzrostu liczności próby. Tabela 5.1 w analogiczny sposób przedstawia analizę dla wzrostu liczby atrybutów w próbie uczące. Wykresy 5.1, 5.2 utworzono na podstawie danych z tabel 5.1 i 5.2. Szczegóły eksperymentu prezentują przykłady 5.3.2 i 5.3.3.

Wniosek 5.3.1 *Analiza wykresów 5.1 i 5.2 wskazuje na liniowy charakter zależności czasu uczenia od liczności próby oraz liczby atrybutów ją opisujących.*

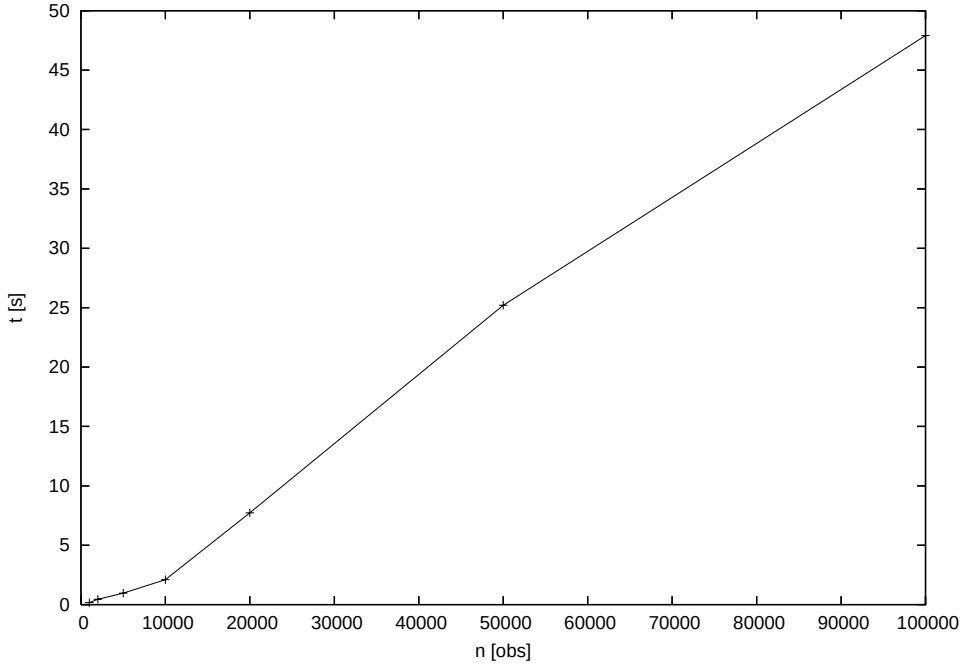
	Liczność	**1 000**	**2 000**	**5 000**	**10 000**	**20 000**	**50 000**	**100 000**
Faza budowy								
	Liczba węzłów	537	1 088	2 558	5 557	9 798	24 421	55 532
	Liczba liści	538	1 089	2 559	5 558	9 799	24 422	55 533
	Głębokość drzewa	16	18	22	27	41	41	52
	Precyzja	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
	Czas budowy	0,16	0,44	0,94	2,08	7,66	25,05	47,68
Faza przycinania								
	Liczba węzłów	231	344	584	92	874	590	541
	Liczba liści	232	345	585	93	875	591	542
	Głębokość drzewa	14	18	21	16	33	36	28
	Precyzja	82,8%	79,4%	76,1%	63,0%	69,0%	63,9%	61,6%
	Czas przycinania	0,01	0,01	0,02	0,02	0,06	0,15	0,23
Uproszczenie drzewa		57,0%	68,4%	77,2%	98,3%	91,1%	97,6%	99,0%
Łączny czas		0,17	0,45	0,96	2,10	7,72	25,20	47,91

Tabela 5.1: Skalowalność SLIQ względem liczność próby uczącej

Przykład 5.3.2 *Szczegóły analizy wpływu wzrostu liczność próby uczącej na czas uczenia.*

```
attributes: 11 events: 1000

Confusion Matrix
        System Answer
    C3  C5  C4  C1  C2  other
C3  202 0   0   0   0   0
C5  0   193 0   0   0   0
C4  0   0   202 0   0   0
C1  0   0   0   196 0   0
C2  0   0   0   0   207 0
other   0   0   0   0   0   0
```

```
attributes: 11 events: 2000

Confusion Matrix
        System Answer
    C5  C1  C3  C4  C2  other
C5  396 0   0   0   0   0
C1  0   392 0   0   0   0
C3  0   0   397 0   0   0
C4  0   0   0   402 0   0
C2  0   0   0   0   413 0
other   0   0   0   0   0   0
```



Rysunek 5.1: Skalowalność SLIQ względem liczność próby uczącej

```
Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 537
Leafs number ..... : 538
Tree height ..... : 16
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 0.161
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.161
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 231
Leafs number ..... : 232
Tree height ..... : 14
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 0.161
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 0.171
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C3  C5  C4  C1  C2  other
C3  348 24  11  12  9   0
C5  22  341 9   11  3   0
C4  29  30  321 15  9   0
C1  26  34  6   322 4   0
C2  27  37  7   19  324 0
other   0   0   0   0   0   0

Accuracy : 0.828
================================================
attributes: 11 events: 5000
```

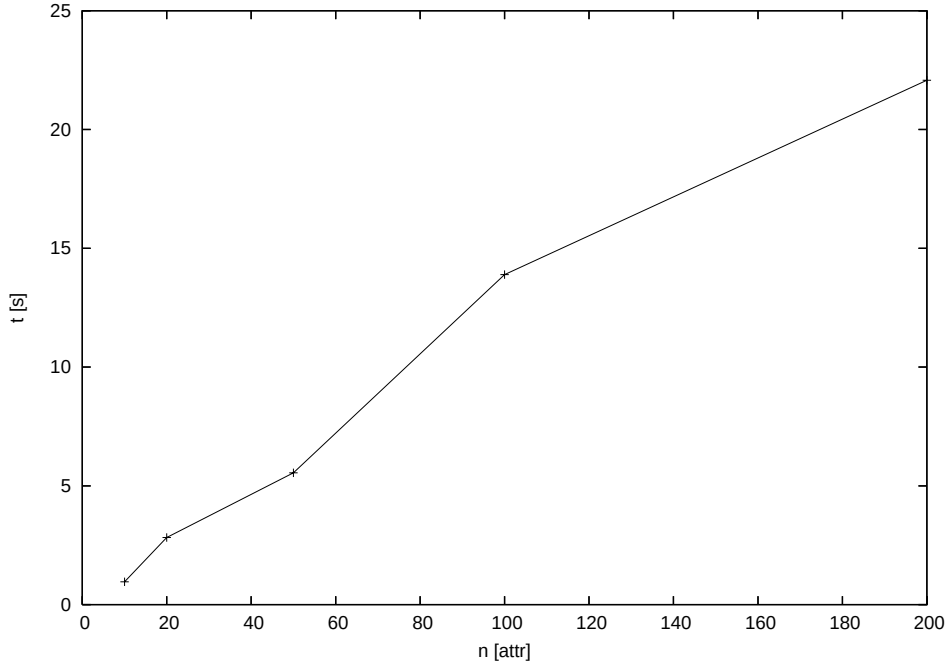
```
Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 1088
Leafs number ..... : 1089
Tree height ..... : 18
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 0.441
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.441
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 344
Leafs number ..... : 345
Tree height ..... : 18
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 0.441
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 0.451
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C5  C1  C3  C4  C2  other
C5  656 32  27  32  45  0
C1  48  614 25  46  51  0
C3  41  42  601 51  59  0
C4  56  42  22  640 44  0
C2  44  41  43  32  666 0
other   0   0   0   0   0   0

Accuracy : 0.79425
================================================
attributes: 11 events: 10000
```

	Liczba atrybutów	**10**	**20**	**50**	**100**	**200**
Faza budowy						
	Liczba węzłów	2 801	1 949	1 651	1 459	1 384
	Liczba liści	2 801	1 950	1 652	1 460	1 385
	Głębokość drzewa	23	22	20	19	20
	Precyzja	100,0%	100,0%	100,0%	100,0%	100,0%
	Czas budowy	0,95	2,81	5,55	13,88	22,06
Faza przycinania						
	Liczba węzłów	296	341	303	397	319
	Liczba liści	297	342	304	398	320
	Głębokość drzewa	19	18	20	19	18
	Precyzja	69,2%	74,1%	73,7%	78,2%	75,8%
	Czas przycinania	0,01	0,01	0,00	0,01	0,01
Uproszczenie drzewa		89,4%	82,5%	81,6%	72,8%	77,0%
Łączny czas		0,96	2,82	5,55	13,89	22,07

Tabela 5.2: Skalowalność SLIQ względem liczby w próbie uczącej

```
Confusion Matrix
        System Answer
    C3  C1  C4  C2  C5  other
C3  1046    0   0   0   0   0
C1  0   994 0   0   0   0
C4  0   0   918 0   0   0
C2  0   0   0   994 0   0
C5  0   0   0   0   1048    0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 2558
Leafs number ..... : 2559
Tree height ..... : 22
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 0.941
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.941
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 584
Leafs number ..... : 585
Tree height ..... : 21
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 0.941
Tree pruning time [s] ..... : 0.02
Total training time [s] ... : 0.96099997
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C3  C1  C4  C2  C5  other
C3  1656    87  155 117 77  0
C1  153 1529    151 66  89  0
C4  135 104 1408    87  102 0
C2  153 103 181 1456    95  0
C5  127 113 181 114 1561    0
other   0   0   0   0   0   0

Accuracy : 0.761
================================================
```

```
Confusion Matrix
        System Answer
    C3  C2  C4  C1  C5  other
C3  1994    0   0   0   0   0
C2  0   2012    0   0   0   0
C4  0   0   1945    0   0   0
C1  0   1   0   2040    0   0
C5  0   0   0   0   2008    0
other   0   0   0   0   0   0

Accuracy : 0.9999
------------- Tree info -------------
Nodes number ..... : 5557
Leafs number ..... : 5558
Tree height ..... : 27
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 2.083
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 2.083
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 92
Leafs number ..... : 93
Tree height ..... : 16
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 2.083
Tree pruning time [s] ..... : 0.02
Total training time [s] ... : 2.103
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C3  C2  C4  C1  C5  other
C3  2541    184 515 219 529 0
C2  384 2315    453 271 601 0
C4  389 172 2548    229 552 0
C1  425 174 503 2461    519 0
C5  342 202 497 244 2731    0
other   0   0   0   0   0   0

Accuracy : 0.6298
================================================
```



Rysunek 5.2: Skalowalność SLIQ względem liczby atrybutów w próbie uczącej

```
attributes: 11 events: 20000

Confusion Matrix
        System Answer
    C1  C2  C4  C3  C5  other
C1  4056    0   0   0   0   0
C2  0   4097    0   0   0   0
C4  0   0   4018    0   0   0
C3  0   0   0   3960    0   0
C5  0   0   0   0   3869    0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 9798
Leafs number ..... : 9799
Tree height ..... : 41
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 7.661
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 7.661
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 874
Leafs number ..... : 875
Tree height ..... : 33
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 7.661
Tree pruning time [s] ..... : 0.061
Total training time [s] ... : 7.7219996
================================================
Testing... Done!
Confusion Matrix
        System Answer
```

```
attributes: 11 events: 50000

Confusion Matrix
        System Answer
    C1  C4  C3  C2  C5  other
C1  9964    0   0   0   0   0
C4  0   9926    0   0   0   0
C3  0   0   9904    0   0   0
C2  0   0   0   10134   0   0
C5  0   0   0   0   10072   0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 24421
Leafs number ..... : 24422
Tree height ..... : 41
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 25.046
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 25.046
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 590
Leafs number ..... : 591
Tree height ..... : 36
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 25.046
Tree pruning time [s] ..... : 0.151
Total training time [s] ... : 25.196999
================================================
Testing... Done!
Confusion Matrix
        System Answer
```

```
    C1  C2  C4  C3  C5  other
C1  5835    595 542 461 679 0
C2  747 5620    565 521 741 0
C4  747 597 5413    526 753 0
C3  724 616 614 5285    681 0
C5  728 581 539 452 5438    0
other   0   0   0   0   0   0

Accuracy : 0.689775
================================================
attributes: 11 events: 100000

Confusion Matrix
        System Answer
    C5  C4  C2  C1  C3  other
C5  19884   0   0   0   0   0
C4  0   19965   0   0   0   0
C2  0   0   20065   0   0   0
C1  0   0   0   20064   0   0
C3  0   0   0   0   20022   0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 55532
Leafs number ..... : 55533
Tree height ..... : 52
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 47.679
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 47.679
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 541
Leafs number ..... : 542
Tree height ..... : 28
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 47.679
Tree pruning time [s] ..... : 0.23
Total training time [s] ... : 47.909
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C5  C4  C2  C1  C3  other
C5  30226   2180    3202    3254    906 0
C4  9581    22917   3224    3303    905 0
C2  9514    2217    24149   3381    869 0
C1  9432    2303    3056    24418   919 0
C3  9753    2243    3046    3434    21568   0
other   0   0   0   0   0   0

Accuracy : 0.61639
```

```
    C1  C4  C3  C2  C5  other
C1  12439   2069    1629    2201    1590    0
C4  1540    13052   1568    2156    1536    0
C3  1495    2069    12523   2196    1525    0
C2  1571    2082    1570    13414   1631    0
C5  1521    2166    1700    2250    12507   0
other   0   0   0   0   0   0

Accuracy : 0.63935
================================================
attributes: 11 events: 200000

Confusion Matrix
        System Answer
    C4  C1  C2  C5  C3  other
C4 39947    0   0   0   0   0
C1 0    39983   0   0   0   0
C2 0    0   40044   0   0   0
C5 0    0   0   40041   0   0
C3 0    0   0   0   39985   0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 94988
Leafs number ..... : 94989
Tree height ..... : 74
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 195.541
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 195.541
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 947
Leafs number ..... : 948
Tree height ..... : 53
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 195.541
Tree pruning time [s] ..... : 0.932
Total training time [s] ... : 196.473
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C4  C1  C2  C5  C3  other
C4 51075    4791    6419    7827    9782    0
C1 9088     46473   6348    8099    9958    0
C2 9281     4642    48320   7897    9948    0
C5 8945     4646    6561    49953   9977    0
C3 9222     4645    6258    8054    51791   0
other   0   0   0   0   0   0

Accuracy : 0.61903
```

Przykład 5.3.3 *Szczegóły analizy wpływu wzrostu liczby atrybutów próbie uczącej na czas uczenia.*

```
attributes: 11 events: 5000

Confusion Matrix
        System Answer
    C1  C3  C2  C4  C5  other
C1  991 0   0   0   0   0
C3  0   979 0   0   0   0
C2  0   0   1043    0   0   0
C4  0   0   0   937 0   0
C5  0   0   0   0   1050    0
```

```
attributes: 21 events: 5000

Confusion Matrix
        System Answer
    C4  C3  C2  C5  C1  other
C4  951 0   0   0   0   0
C3  0   1063    0   0   0   0
C2  0   0   1012    0   0   0
C5  0   0   0   968 0   0
C1  0   0   0   0   1006    0
```

```
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 2801
Leafs number ..... : 2802
Tree height ..... : 23
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 0.952
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 0.952
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 296
Leafs number ..... : 297
Tree height ..... : 19
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 0.952
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 0.962
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C1  C3  C2  C4  C5  other
C1  1386    147 126 62  261 0
C3  162 1376    122 64  234 0
C2  164 184 1444    49  245 0
C4  142 156 155 1190    231 0
C5  161 177 155 87  1520    0
other   0   0   0   0   0   0

Accuracy : 0.6916
================================================
attributes: 51 events: 5000

Confusion Matrix
        System Answer
    C4  C1  C3  C5  C2  other
C4  1008    0   0   0   0   0
C1  0   990 0   0   0   0
C3  0   0   1030    0   0   0
C5  0   0   0   956 0   0
C2  0   0   0   0   1016    0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 1651
Leafs number ..... : 1652
Tree height ..... : 20
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 5.548
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 5.548
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 303
Leafs number ..... : 304
Tree height ..... : 20
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 5.548
Tree pruning time [s] ..... : 0.0
```

```
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 1949
Leafs number ..... : 1950
Tree height ..... : 22
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 2.814
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 2.814
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 341
Leafs number ..... : 342
Tree height ..... : 18
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 2.814
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 2.824
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C4  C3  C2  C5  C1  other
C4  1394    116 155 144 93  0
C3  134 1551    166 172 103 0
C2  137 115 1533    153 86  0
C5  111 114 137 1469    105 0
C1  127 89  168 169 1459    0
other   0   0   0   0   0   0

Accuracy : 0.7406
================================================
attributes: 101 events: 5000

Confusion Matrix
        System Answer
    C1  C4  C3  C2  C5  other
C1  1054    0   0   0   0   0
C4  0   982 0   0   0   0
C3  0   0   1018    0   0   0
C2  0   0   0   992 0   0
C5  0   0   0   0   954 0
other   0   0   0   0   0   0

Accuracy : 1.0
------------- Tree info -------------
Nodes number ..... : 1459
Leafs number ..... : 1460
Tree height ..... : 19
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 13.88
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 13.88
================================================
MDL Pruning... Done!
------------- Tree info -------------
Nodes number ..... : 397
Leafs number ..... : 398
Tree height ..... : 19
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 13.88
Tree pruning time [s] ..... : 0.01
```

```
Total training time [s] ... : 5.548
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C4  C1  C3  C5  C2  other
C4  1464    145 134 129 144 0
C1  133 1508    99  116 124 0
C3  136 153 1521    126 124 0
C5  143 131 110 1395    133 0
C2  106 169 114 160 1483    0
other   0   0   0   0   0   0

Accuracy : 0.7371
================================================
attributes: 201 events: 5000

Confusion Matrix
        System Answer
    C4  C2  C1  C5  C3  other
C4  1020    0   0   0   0   0
C2  0   993 0   0   0   0
C1  0   0   973 0   0   0
C5  0   0   0   1004    0   0
C3  0   0   0   0   1010    0
other   0   0   0   0   0   0

Accuracy : 1.0
------------ Tree info -------------
Nodes number ..... : 1384
Leafs number ..... : 1385
Tree height ..... : 20
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 22.062
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 22.062
================================================
MDL Pruning... Done!
------------ Tree info -------------
Nodes number ..... : 319
Leafs number ..... : 320
Tree height ..... : 18
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 22.062
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 22.072
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C4  C2  C1  C5  C3  other
C4  1484    143 133 154 126 0
C2  100 1527    93  143 123 0
C1  94  123 1486    129 114 0
C5  84  166 83  1567    108 0
C3  100 173 80  152 1515    0
other   0   0   0   0   0   0

Accuracy : 0.7579
```

```
Total training time [s] ... : 13.89
================================================
Testing... Done!
Confusion Matrix
        System Answer
    C1  C4  C3  C2  C5  other
C1  1721    120 93  98  76  0
C4  156 1573    70  106 59  0
C3  172 144 1544    108 68  0
C2  185 94  101 1534    70  0
C5  140 121 86  118 1443    0
other   0   0   0   0   0   0

Accuracy : 0.7815
================================================
attributes: 301 events: 5000

Confusion Matrix
        System Answer
   C1  C3  C4  C5  C2  other
C1 999 0   0   0   0   0
C3 0   1024    0   0   0   0
C4 0   0   995 0   0   0
C5 0   0   0   998 0   0
C2 0   0   0   0   984 0
other  0   0   0   0   0   0

Accuracy : 1.0
------------ Tree info -------------
Nodes number ..... : 1339
Leafs number ..... : 1340
Tree height ..... : 21
Tree built ..... : true
Tree pruned ..... : false
Tree building time [s] .... : 40.398
Tree pruning time [s] ..... : 0.0
Total training time [s] ... : 40.398
================================================
MDL Pruning... Done!
------------ Tree info -------------
Nodes number ..... : 127
Leafs number ..... : 128
Tree height ..... : 16
Tree built ..... : true
Tree pruned ..... : true
Tree building time [s] .... : 40.398
Tree pruning time [s] ..... : 0.01
Total training time [s] ... : 40.407997
================================================
Testing... Done!
Confusion Matrix
        System Answer
   C1  C3  C4  C5  C2  other
C1 1421    178 74  201 124 0
C3 162 1483    66  170 167 0
C4 171 189 1286    187 157 0
C5 140 173 68  1455    160 0
C2 140 157 83  191 1397    0
other  0   0   0   0   0   0

Accuracy : 0.7042
```

Dodatek A

Topologia, prawdopodobieństwo i miara

A.1. Przestrzeń topologiczna

Niech Ω będzie dowolnym niepustym zbiorem.

Definicja A.1.1 *Rodzinę $\mathfrak{T}$ podzbiorów zbioru Ω nazywamy topologią w zbiorze Ω jeżeli spełnione są następujące warunki:*

(T_1) $\emptyset \in \mathfrak{T}$, $\Omega \in \mathfrak{T}$.

(T_2) $(U \in \mathfrak{T}) \wedge (V \in \mathfrak{T}) \Rightarrow (U \cap V \in \mathfrak{T})$,

(T_3) $(\forall\ s \in S \quad U_s \in T) \Rightarrow \left(\bigcup_{s \in S} U_s \in \mathfrak{T} \right)$ *dla dowolnego zbioru S.*

Topologia jest uogólnieniem[1] pojęcia zbioru otwartego. Elementy rodziny $\mathfrak{T}$ nazywamy zbiorami otwartymi w Ω, a parę $(\Omega, \mathfrak{T})$ nazywamy przestrzenią topologiczną. Zauważmy, że topologia jest rodziną zamkniętą na skończony iloczyn, oraz dowolną sumę zbiorów.

Przykład A.1.1 *W zbiorze Ω wyróżnia się dwie skrajne topologie:*

- $\{\emptyset, \Omega\}$ *- topologia minimalna,*
- 2^{Ω} *- topologia dyskretna.*

Przykład A.1.2 *Zakładając, że $\Omega \neq \emptyset$ i $\omega \in \Omega$ można podać dwa kolejne przykłady topologii w Ω:*

- $\{\emptyset\} \cup \{U \subseteq \Omega\ :\ \omega \in U\}$
- $\{\emptyset\} \cup \{U \subseteq \Omega\ :\ \omega \notin U\}$

Twierdzenie A.1.1 *Niech $(\Omega, \mathfrak{T})$ będzie przestrzenią topologiczną, a S ustalonym podzbiorem zbioru Ω. Rodzina $\mathfrak{T}_S$ podzbiorów zbioru S określona następująco:*

$$\mathfrak{T}_S := \{S \cap G\ :\ G \in \mathfrak{T}\}$$

jest topologią w S.

Wniosek A.1.1 *W każdym podzbiorze przestrzeni topologicznej można wprowadzić topologię.*

Topologię $\mathfrak{T}_S$ w S nazywamy topologią indukowaną przez topologię $\mathfrak{T}$ w Ω.

[1] wykracza znacznie poza pojęcie otwartości zbioru w przestrzeniach metrycznych

A.2. σ-ciało i σ-ciało zbiorów Borela

Definicja A.2.1 *Rodzinę $\mathfrak{F}$ podzbiorów zbioru Ω nazywamy σ-ciałem podzbiorów zbioru Ω (σ-ciałem w Ω) jeżeli:*

(σ_1) $\emptyset \in \mathfrak{F}$,

(σ_2) $(A \in \mathfrak{F}) \Rightarrow (\Omega \setminus A \in \mathfrak{F})$,

(σ_3) $\left(A_i \in \mathfrak{F} \quad \text{dla} \quad i = 1, 2, \dots\right) \Rightarrow \left(\bigcup_{i=1}^{\infty} A_i \in \mathfrak{F}\right)$.

Wniosek A.2.1 *Przeliczalnie addytywne ciało zbiorów jest σ-ciałem.*

Przykład A.2.1 *Rodziny $\{\emptyset, \Omega\}$, 2^{Ω} są σ-ciałami podzbiorów zbioru Ω.*

Każde σ-ciało zbiorów jest zamknięte ze względu na dopełnienie zbioru i przeliczalną sumę zbiorów. Dalej przekonamy się, że σ-ciała posiadają znacznie więcej własności.

Twierdzenie A.2.1 *Jeżeli $\mathfrak{F}$ jest σ-ciałem podzbiorów zbioru Ω, to:*

(w_1) $\Omega \in \mathfrak{F}$,

(w_2) $\left(A_i \in \mathfrak{F} \quad \text{dla} \quad i = 1, 2, \dots, n\right) \Rightarrow \left(\bigcup_{i=1}^{n} A_i \in \mathfrak{F}\right)$,

(w_3) $\left(A_i \in \mathfrak{F} \quad \text{dla} \quad i = 1, 2, \dots\right) \Rightarrow \left(\bigcap_{i=1}^{\infty} A_i \in \mathfrak{F}\right)$,

(w_4) $\left(A_i \in \mathfrak{F} \quad \text{dla} \quad i = 1, 2, \dots, n\right) \Rightarrow \left(\bigcap_{i=1}^{n} A_i \in \mathfrak{F}\right)$,

(w_5) $(A \in \mathfrak{F}) \wedge (B \in \mathfrak{F}) \Rightarrow (A \setminus B \in \mathfrak{F})$,

(w_6) *jeżeli $\{\mathfrak{F}_\alpha\}_{\alpha \in I}$ jest rodziną σ-ciał w zbiorze Ω, to przecięcie $\bigcap_{\alpha \in I} \mathfrak{F}_\alpha$ jest σ-ciałem w Ω.*

Twierdzenie A.2.2 *Dla każdej rodziny $\mathfrak{R}$ podzbiorów zbioru Ω istnieje dokładnie jedno σ-ciało $\mathfrak{F}_0$ w Ω takie, że:*

1. $\mathfrak{R} \subseteq \mathfrak{F}_0$,
2. $\mathfrak{F}_0 \subseteq \mathfrak{F}$ *dla każdego σ-ciała $\mathfrak{F}$ w Ω.*

$\mathfrak{F}_0$ jest najmniejszym σ-ciałem w Ω zawierającym rodzinę $\mathfrak{R}$, nazywamy je σ-ciałem generowanym przez rodzinę $\mathfrak{R}$.

Definicja A.2.2 *Jeżeli $(\Omega, \mathfrak{T})$ jest przestrzenią topologiczną z topologią $\mathfrak{T}$, to σ-ciałem borelowskim w przestrzeni $(\Omega, \mathfrak{T})$ nazywamy σ-ciało generowane przez topologię $\mathfrak{T}$.*

Definicja A.2.3 *Przestrzenią mierzalną nazywamy parę $(\Omega, \mathfrak{F})$, gdzie Ω dowolny zbiór, a $\mathfrak{F}$ σ-ciało w Ω.*

A.3. Miara i miara probabilistyczna

$(\Omega, \mathfrak{F})$ - przestrzeń mierzalna.

Definicja A.3.1 *Funkcję $\mu : \mathfrak{F} \to \tilde{\mathbb{R}}$ nazywamy miarą, jeżeli spełnione są następujące warunki:*

(μ_1) $\forall\ A \in \Omega,\ \mu(A) \geqslant 0$

(μ_2) $\mu(\emptyset) = 0$

(μ_3) *μ jest przeliczalnie addytywną funkcją zbioru, tj. dla każdego ciągu zbiorów $A_i \in \mathfrak{F}$ $(i = 1, 2, \ldots)$*

$$(A_i \cap A_j = \emptyset \quad \text{dla} \quad i \neq \jmath) \Rightarrow \left(\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} \mu(A_i)\right)$$

Mówimy, że zbiory należące do σ-ciała $\mathfrak{F}$ są μ-mierzalne (lub krótko - mierzalne). Miara jest uogólnieniem pojęć takich jak długość, pole powierzchni, objętość zbioru[2]. Trzy aksjomaty podane w definicji A.3.1 implikują kilka kolejnych bardzo ważnych własności miary.

Twierdzenie A.3.1 *Niech $(\Omega, \mathfrak{F})$ będzie przestrzenią mierzalną, a μ miarą określoną na σ-ciele $\mathfrak{F}$. Wtedy dla dowolnych $A_i \in \mathfrak{F}$ $(i = 1, 2, \ldots, n)$ i $A, B \in \mathfrak{F}$ zachodzi:*

(w_1) $(A_i \cap A_j = \emptyset \quad \text{dla} \quad i \neq \jmath) \Rightarrow \left(\mu\Big(\bigcup_{i=1}^{n} A_i\Big) = \sum_{i=1}^{n} \mu(A_i)\right)$

(w_2) $(A \subseteq B) \Rightarrow (\mu(A) \leqslant \mu(B))$

(w_3) $(A \subseteq B) \wedge (\mu(B) < +\infty) \Rightarrow (\mu(B \setminus A) = \mu(B) - \mu(A))$

(w_4) $\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) \leqslant \sum_{i=1}^{\infty} \mu(A_i)$

(w_5) $\mu\Big(\bigcup_{i=1}^{n} A_i\Big) \leqslant \sum_{i=1}^{n} \mu(A_i)$

(w_6) $(A_i \subseteq A_{i+1}) \Rightarrow \left(\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \lim_{i \to \infty} \mu(A_i)\right)$

(w_7) $(\mu(A_1) < +\infty) \wedge (A_i \subseteq A_{i+1}) \Rightarrow \left(\mu\Big(\bigcap_{i=1}^{\infty} A_i\Big) = \lim_{i \to \infty} \mu(A_i)\right)$

Dowody twierdzeń A.2.1, A.2.2, A.3.1 można znaleźć w [8] (rozdział IV, §42, 2).

Definicja A.3.2 *Każdą miarę P w przestrzeni mierzalnej $(\Omega, \mathfrak{F})$ spełniającą warunek:*

$$P(\Omega) = 1$$

nazywamy miarą probabilistyczną (prawdopodobieństwem).

[2]W przestrzeniach $\mathbb{R}^n$ definiuje się najczęściej miarę Lebesgue'a

Definicja A.3.3 *Przestrzenią probabilistyczną nazywamy dowolną trójkę* $(\Omega, \mathfrak{F}, P)$*, gdzie* $(\Omega, \mathfrak{F})$ *jest przestrzenią mierzalną, a* P *miarą probabilistyczną w* $(\Omega, \mathfrak{F})$*.*

Zbiór Ω nazywamy zbiorem zdarzeń elementarnych. Miara probabilistyczna posiada wszystkie własności miary wymienione w twierdzeniu A.3.1. Dalej przedstawimy cechy miary probabilistycznej, które nie zachodzą w ogólnej sytuacji.

Twierdzenie A.3.2 *Jeżeli* (Ω, F, P) *jest przestrzenią probabilistyczną i* $A, B \in \mathfrak{F}$*, to:*

(P_1) $P(A) \leqslant 1$

(P_2) $P(A) = 1 - P(\Omega \setminus A)$

(P_3) $(A \subseteq B) \Rightarrow (P(B \setminus A) = P(B) - P(A))$

(P_4) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Dowód twierdzenia A.3.2 można znaleźć w [10] (rozdział I, §1.1).

Definicja A.3.4 *Prawdopodobieństwo zdarzenia* $A \in \mathfrak{F}$ *pod warunkiem zdarzenia* $B \in \mathfrak{F}$ *o ile* $P(B) > 0$ *definiujemy jako:*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Twierdzenie A.3.3 (Bayesa) *Dla takich zdarzeń* $A, B \in \mathfrak{F}$*, że* $P(A) > 0$ *i* $P(B) > 0$ *zachodzi:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Twierdzenie A.3.4 (Bayesa) *Dla takich parami rozłącznych zdarzeń* $A_1, A_2, \ldots$*, że* $P(A_n) > 0$*,* $n = 1, 2, \ldots$ *i* $\bigcup_n A_n = \Omega$*, oraz dla każdego takiego zdarzenia* $B \in \mathfrak{F}$*, że* $P(B) > 0$ *zachodzi:*

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{\sum_n P(A_n)P(B|A_n)}$$

A.4. Funkcje mierzalne

Zakładamy, że $(\Omega, \mathfrak{F})$ jest przestrzenią mierzalną. Dodatkowo niech A będzie dowolnym podzbiorem zbioru Ω.

Definicja A.4.1 *Funkcję* $f : A \to \tilde{\mathbb{R}}$ *nazywamy funkcją mierzalną (względem* σ*-ciała* $\mathfrak{F}$*) jeżeli* $A \in \mathfrak{F}$ *oraz:*

$$\forall\ a \in \mathbb{R} \quad \{x\ :\ f(x) < a\} \in \mathfrak{F}$$

Przykładem funkcji mierzalnej jest funkcja stała na zbiorze mierzalnym. Istnieją cztery równoważne definicje mierzalności funkcji.

Twierdzenie A.4.1 *Niech* $f : A \to \tilde{\mathbb{R}}$*, gdzie* $A \in \mathfrak{F}$*. Następujące warunki są równoważne:*

(m_1) *f jest funkcją mierzalną,*

(m_2) $\forall\ a \in \mathbb{R} \quad \{x\ :\ f(x) \leqslant a\} \in \mathfrak{F}$

$(m_3)\ \forall\ a \in \mathbb{R} \quad \{x\ :\ f(x) > a\} \in \mathfrak{F}$

$(m_4)\ \forall\ a \in \mathbb{R} \quad \{x\ :\ f(x) \geqslant a\} \in \mathfrak{F}$

Funkcje mierzalne generują bardzo ważną klasę zbiorów mierzalnych[3]. Zamieszczamy poniżej charakterystykę zbiorów mierzalnych generowanych przez funkcje mierzalne.

Twierdzenie A.4.2 *Dla dowolnych mierzalnych funkcji $f, g : A \to \tilde{\mathbb{R}}$, oraz dowolnych $a, b \in \mathbb{R}$ zachodzą następujące własności:*

(w_1) $\{x\ :\ f(x) < +\infty\} \in \mathfrak{F}$,
$\{x\ :\ f(x) > -\infty\} \in \mathfrak{F}$,

(w_2) $\{x\ :\ a < f(x) < b\} \in \mathfrak{F}$,
$\{x\ :\ a \leqslant f(x) < b\} \in \mathfrak{F}$,
$\{x\ :\ a < f(x) \leqslant b\} \in \mathfrak{F}$,
$\{x\ :\ a \leqslant f(x) \leqslant b\} \in \mathfrak{F}$,

(w_3) $\{x\ :\ f(x) = a\} \in \mathfrak{F}$,
$\{x\ :\ f(x) \neq a\} \in \mathfrak{F}$,

(w_4) $\{x\ :\ f(x) < g(x)\} \in \mathfrak{F}$,
$\{x\ :\ f(x) \leqslant g(x)\} \in \mathfrak{F}$,
$\{x\ :\ f(x) = g(x)\} \in \mathfrak{F}$,
$\{x\ :\ f(x) \neq g(x)\} \in \mathfrak{F}$.

Twierdzenie A.4.3 *Obcięcie funkcji mierzalnej do niepustego zbioru mierzalnego jest funkcją mierzalną.*

Dowody twierdzeń A.4.1, A.4.2, A.4.3 można znaleźć w [8] (rozdział VI, §44).

[3]W dalszych rozważaniach najczęściej wykorzystywać będziemy własność przechodzenia mierzalności funkcji na mierzalność jej przeciwobrazów (ich pewnej klasy).

Dodatek B

Teoria grafów

B.1. Grafy

Definicja B.1.1 *Parę $G =< V, E >$ nazywamy grafem jeżeli V jest dowolnym niepustym zbiorem skończonym oraz $E \subseteq P_2(V) := \{A \subseteq V \ : \ |A| = 2\}$. Zbiór V nazywamy zbiorem wierzchołków, zbiór E nazywamy zbiorem krawędzi grafu G.*

Definicja B.1.2 *Parę $G =< V, E >$ nazywamy grafem zorientowanym jeżeli V jest dowolnym niepustym zbiorem skończonym oraz $E \subseteq V \times V$.*

Każdy element $v \in V$ nazywamy wierzchołkiem grafu G. Krawędzią grafu G łączącą wierzchołki $v_1, v_2 \in V$ nazywamy parę $e = \{v_1, v_2\} \in E$.

Definicja B.1.3 *Podgrafem grafu $G =< V, E >$ nazywamy dowolny graf $H =< W, F >$ taki, że $W \subseteq V$ oraz $F \subseteq E$.*

Definicja B.1.4 *Mówimy, że w grafie $G =< V, E >$ wierzchołek $u \in V$ sąsiaduje z wierzchołkiem v jeżeli $\{v, u\} \in E$. Zbiór $V_G(v) := \{u \in V \ : \ \{v, e\} \in E\}$ nazywamy zbiorem wierzchołków sąsiadujących z wierzchołkiem v, zbiór $E_G(v) := \{e \in E \ : \ v \in e\}$ nazywamy zbiorem krawędzi wierzchołka v.*

Wniosek B.1.1 *Dla dowolnego $v \in V$ zachodzi: $|V_G(v)| = |E_G(v)|$.*

Definicja B.1.5 *Stopniem wierzchołka $v \in V$ grafu $G =< V, E >$ nazywamy liczbę:*

$$\deg_G(v) := |E_G(v)| \tag{B.1}$$

Definicja B.1.6 *Drogą o długości $k-1$ $(k \geqslant 2)$ w grafie $G =< V, E >$ nazywamy dowolny ciąg wierzchołków $(v_1, \ldots, v_k)$ taki, że:*

1. $\{v_i, v_{i+1}\} \in E \quad \text{dla} \quad i = 1, \ldots, k-1$
2. $(i \neq j \quad \text{dla} \quad i, j = 1, \ldots, k-1) \Rightarrow \Big(\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\}\Big)$

Drogę $(v_1, \ldots, v_k)$ oznaczamy $v_1 - v_k$ drogą ($v_1 - v_k$ droga łączy wierzchołki $v_1, v_k \in V$).

Definicja B.1.7 *Drogę $(v_1, \ldots, v_k)$ w grafie $G =< V, E >$ nazywamy drogą prostą jeżeli:*

$$(i \neq j \quad \text{dla} \quad i, j = 1, \ldots, k-1) \Rightarrow (v_i \neq v_j)$$

Definicja B.1.8 *Cyklem w grafie $G =< V, E >$ nazywamy każdą drogę $(v_1, \ldots, v_k)$ o długości większej niż 2 taką, że $v_1 = v_k$.*

Definicja B.1.9 *Graf $G =< V, E >$ nazywamy grafem acyklicznym jeżeli nie posiada cykli.*

Definicja B.1.10 *Graf $G =< V, E >$ nazywamy spójnym wtedy i tylko wtedy, gdy dla każdych $u, v \in V$ istnieje $u - v$ droga w grafie G. Składową grafu G nazywamy każdy maksymalny podgraf spójny grafu G.*

Definicja B.1.11 *Odległością $(dist_G(u, v))$ w grafie $G =< V, E >$ pomiędzy wierzchołkami $u, v \in V$ nazywamy długość najkrótszej $u - v$ drogi w grafie G. W przypadku, gdy wierzchołki u, v leżą w różnych składowych przyjmuje się $dist_G(u, v) = +\infty$.*

Funkcja odległość w grafie jest metryką.

B.2. Drzewa

Definicja B.2.1 *Graf $T =< V, E >$ nazywamy drzewem jeżeli:*

1. *T jest grafem spójnym*
2. *T jest grafem acyklicznym (nie ma cykli).*

Graf, które spełnia tylko warunek 2 nazywamy lasem.

Definicja B.2.2 *Każdy wierzchołek $v \in V$ drzewa $T =< V, E >$ o stopniu $\deg_T(v) = 1$ nazywamy liściem.*

Twierdzenie B.2.1 *Jeżeli $G =< V, E >$ jest grafem to następujące warunki są równoważne:*

1. *graf G jest drzewem*
2. *dla każdych wierzchołków $u, v \in V$ istnieje dokładnie jedna $u - v$ droga w G i jest to droga prosta*
3. *Graf G jest spójny i $|V| = |E| + 1$*
4. *Graf G jest acykliczny i $|V| = |E| + 1$*
5. *Graf G jest acykliczny i dodanie do G nowej krawędzi powoduje powstanie dokładnie jednego cyklu.*

Bibliografia

[1] Jacek Koronacki, Jan Ćwik: *Statystyczne systemy uczące się*, Wydawnictwa Naukowo-Techniczne, Warszawa 2005

[2] Paweł Cichosz: *Systemy uczące się*, Wydawnictwa Naukowo-Techniczne, Warszawa 2000

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone: *Classification and Regression Trees*, Chapman & Hall, New York, NY, 1984

[4] Manish Mehta, Rakesh Agrawal, Jorma Rissanen: *SLIQ: A Fast Scalable Classifier for Data Mining*, Proc. of the Fifth Int'l Conference on Extending Database Technology, Avignon, France, March 1996

[5] John C. Shafer, Rakesh Agrawal, Manish Mehta: *SPRINT: A Scalable Parallel Classifier for Data Mining*, Proc. of the 22th Int'l Conference on Very Large Databases, Mumbai (Bombay), India, Sept. 1996

[6] Manish Mehta, Jorma Rissanen, Rakesh Agrawal: *MDL-based Decision Tree Pruning*, Proc. of the 1st Int'l Conference on Knowledge Discovery in Databases and Data Mining, Montreal, Canada, August, 1995

[7] Kenneth Ross, Charles R. B. Wright : *Matematyka dyskretna*, Państwowe Wydawnictwo Naukowe, Styczeń 2005

[8] Witold Kołodziej: *Analiza matematyczna*, Państwowe Wydawnictwo Naukowe, Warszawa 1978

[9] Kazimierz Kuratowski: *Wstęp do teorii mnogości i topologii*, wydanie piąte, Państwowe Wydawnictwo Naukowe, Warszawa 1972

[10] Jacek Jakubowski, Rafał Sztencel: *Wstęp do teorii prawdopodobieństwa*, Wydanie II, SCRIPT, Warszawa 2001

[11] Patrick Billingsley: *Prawdopodobieństwo i miara*, Państwowe Wydawnictwo Naukowe, Warszawa 1987

Mariusz Gromada
Nr albumu 174094

Warszawa, 31 stycznia 2006

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem „Drzewa Klasyfikacyjne - ich budowa, problemy złożoności i skalowalności", której promotorem jest Prof. dr hab. Jacek Koronacki wykonałam samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Mariusz Gromada