Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Modele inteligencji obliczeniowej dla zadań klasyfikacji danych: metody Bayesowskie

Jakub P. Piątkowski

Praca inżynierska wykonana
w Katedrze Informatyki Stosowanej
Wydziału FAiIS UMK
pod kierunkiem
dra Krzysztofa Grąbczewskiego

Toruń 2004

Spis treści

Wstęp

Za przykład problemu klasyfikacji danych służyć może chory zgłaszający się do lekarza z opisem objawów oraz wynikami badań. Jego stan może być przedstawiony jako zbiór cech o wartościach liczbowych (jak poziom cholesterolu lub ciśnienie krwi) i symbolicznych (np. logicznych: zawroty głowy występują lub nie). Na podstawie tych danych stawiana jest diagnoza. Pacjent zostaje zaliczony do grupy ludzi zdrowych, chorych na określoną chorobę, mających szansę na szybkie wyzdrowienie, lub wymagających długiej kuracji itd. Ten prosty przykład nie odzwierciedla skali zadań spotykanych w klasyfikacji.

Ilość gromadzonych danych wzrasta, a ludzkie zdolności ich przetwarzania — nie. Rośnie również stopień złożoności danych; najczęściej leżą one w przestrzeni wielowymiarowej, przekraczającej ludzką wyobraźnię przestrzenną. Na dodatek klasyfikacja danych przez ekspertów jest bardzo kosztowna. Pojawia się więc konieczność zautomatyzowania tego procesu. Zastosowanie komputerów w połączeniu z metodami inteligencji obliczeniowej daje nadzieję zarówno na pokonanie trudności wynikających z rozmiarów baz danych, jak i na uzyskanie rezultatów niemożliwych do osiągnięcia dla człowieka.

GhostMiner jest przykładem oprogramowania wychodzącego na przeciw tym potrzebom. Udostępnia on wiele narzędzi do budowania złożonych modeli, jak i do oceny ich dokładności. Gotowe modele mogą następnie służyć do analizy danych, w tym do wykrywania niewidocznych na pierwszy rzut oka zależności i do klasyfikacji danych. Przedmiotem tej pracy było dodanie do istniejącego programu *GhostMiner* modułu implementującego naiwny klasyfikator Bayesowski.

Rozdział 1

Klasyfikacja danych

1.1 Podstawowe pojęcia

Naiwny klasyfikator Bayesowski jest przykładem modelu klasyfikującego dane. Innymi słowy, jego zadanie polega na określeniu kategorii, do jakiej zalicza się brany pod uwagę przypadek.

Porcje danych reprezentujące poszczególne przypadki zawierają wartości dobrze ustalonych typów (całkowite, rzeczywiste, logiczne, symboliczne itp) w określonej kolejności. Możemy zatem mówić o nich jako o *wektorach*. Poszczególne składowe takiego wektora nazywamy *cechami* bądź *atrybutami*. Będziemy oznaczać je przez x_k, a cały wektor — przez $x = [x_1, x_2, \ldots, x_r]$, gdzie r to liczba atrybutów. Jeżeli cecha k jest typu symbolicznego, przyjmuje l_k różnych wartości. Wektory możemy również traktować jako punkty w przestrzeni cech.

Zbiór wszystkich wektorów jest podzielony na podzbiory ze względu na interesujące nas własności. Te podzbiory to *klasy*. Rozważamy problemy klasyfikacji, w których klasy są rozłączne, a ich suma jest równa całej przestrzeni. Możemy zatem zakładać, że każdy punkt należy do dokładnie jednej klasy. Zbiór wszystkich klas będziemy oznaczać jako C, a pojedynczą klasę jako $c \in C$.

Zadaniem *klasyfikatorów* jest przyporządkowanie punktów do odpowiednich klas. Zazwyczaj nie wiemy jednak, jaki dokładnie obszar w przestrzeni cech obejmuje dana klasa. Jedyne, co mamy do dyspozycji, to zbiór wektorów, których przynależność klasową znamy. Ta próbka populacji wszystkich możliwych wektorów jest określana jako *zbiór treningowy*. Należące do niego punkty oznaczane będą jako $x^{(j)}$.

Liczebność całego zbioru treningowego będzie zapisywana jako n. Natomiast liczba wektorów z tego zbioru spełniających określony warunek oznaczana będzie przez $n_{warunek}$, na przykład liczebność klasy c to $n_{x^{(j)} \in c}$.

Klasyfikatory zaliczają nowe punkty do klas na podstawie wiedzy zawartej w zbiorze treningowym. Jego reprezentatywność (liczebność i rozkład punktów) bardzo mocno wpływa na jakość klasyfikacji. Najczęściej przed rozpoczęciem klasyfikacji model bada zależności występujące w zbiorze treningowym, co określamy jako *proces uczenia*.

1.2 Prawdopodobieństwo

Naiwny klasyfikator Bayesowski przybliża prawdopodobieństwa przynależności wektora do poszczególnych klas. Następnie przypisuje go do najbardziej prawdopodobnej klasy.

Prawdopodobieństwo tego, że jakikolwiek nieznany wektor będzie należał do klasy c (co oznaczamy jako $P(x \in c)$) nazywamy prawdopodobieństwem „a priori" tej klasy. Jeżeli natomiast znamy wartości cech wektora, posługujemy się prawdopodobieństwem „a posteriori". Jest to prawdopodobieństwo warunkowe, które możemy zapisać jako: $P(x \in c \mid x = [x_1, x_2, \ldots, x_r])$. Jest to prawdopodobieństwo tego, że punkt x należy do klasy c, jeżeli wiemy, że jego cechy mają wartości $x_1, x_2, \ldots, x_r$.

W dalszej części pracy, w celu skrócenia zapisu, przyjęta zostanie następująca konwencja: $x \in c$ będzie wyrażane przez c, $x = [x_1, x_2, \ldots, x_r]$ — przez $x_1, x_2, \ldots x_r$, a $x_k^{(j)} = x_k$ — przez x_k. Przykładowo, liczba wektorów ze zbioru treningowego należących do klasy c, których k-ta cecha ma wartość taką samą, jak w wektorze x, będzie zapisana jako $n_{c \wedge x_k}$ zamiast $n_{x^{(j)} \in c \wedge x_k^{(j)} = x_k}$.

Używając powyższych oznaczeń, najbardziej prawdopodobna (np) klasa to:

$$c_{np} = \arg\max_{c \in C} P(c \mid x_1, x_2, \ldots, x_r) \tag{1.1}$$

1.3 Wzór Bayesa

Znany w matematyce wzór Bayesa[1] dotyczy prawdopodobieństwa warunkowego. Niech A i B oznaczają obserwację zdarzeń losowych. Przez $P(A)$ oznaczmy prawdopodobieństwo zajścia zdarzenia A. Natomiast $P(A \mid B)$ to

[1] Thomas Bayes (1702-63), matematyk angielski

prawdopodobieństwo zajścia A pod warunkiem, że zaszło B. Matematycznie można to zapisać tak:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{1.2}$$

Wyrażenie $A \cap B$ oznacza jednoczesne zajście A i B. Wzór Bayesa pozwala wyrazić to prawdopodobieństwo inaczej:

$$P(A \mid B) = \frac{P(B \mid A)P(B)}{P(A)} \tag{1.3}$$

Wyrażenie określające najbardziej prawdopodobną klasę (1.1) można, używając wzoru Bayesa (1.3), przekształcić do postaci następującej:

$$c_{np} = \arg\max_{c \in C} \frac{P(x_1, x_2, \ldots, x_r \mid c)P(c)}{P(x_1, x_2, \ldots, x_r)}$$

Ponieważ mianownik nie zależy od klasy, nie zmienia wyniku klasyfikacji. Pomijając go otrzymujemy ostatecznie:

$$c_{np} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_r \mid c)P(c) \tag{1.4}$$

Jeżeli prawdopodobieństwa występujące we wzorze (1.4) są znane, bądź jesteśmy je w stanie oszacować, możemy stosować go bezpośrednio do klasyfikacji. Model działający według tego wzoru nosi nazwę optymalnego klasyfikatora Bayesowskiego.

Rozdział 2

Naiwny klasyfikator Bayesowski

2.1 Założenia

Zazwyczaj prawdopodobieństwa warunkowe występujące we wzorze (1.4) nie są znane. Możemy je jedynie szacować na podstawie znajomości zbioru treningowego. Osiągnięcie dokładnego oszacowania wymagałoby olbrzymiego zbioru danych, ponieważ na klasyfikację nowego punktu wpływałyby jedynie punkty położone w jego bezpośrednim otoczeniu w przestrzeni cech.

Metoda prostego zliczania (patrz podrozdział 2.2.1) działająca w oparciu o równanie (1.4) zliczałaby wektory identyczne z testowanym w ramach różnych klas. Odpowiednie prawdopodobieństwa byłyby wówczas przybliżane przez:

$$P(x_1, x_2, \ldots, x_r \mid c) = \frac{n_{x_1,x_2,\ldots,x_r \wedge c}}{n_c}$$

Zatem w przypadku danych bez niejednoznaczności[1], rola klasyfikatora działającego w oparciu o wzór (1.4) sprowadzałaby się do zapamiętywania napotkanych przypadków. Oznaczałoby to całkowitą niezdolność do generalizacji.

Taki algorytm nie byłby zbyt użyteczny. Z tego względu konieczna jest modyfikacja wzoru (1.4). Naiwny klasyfikator Bayesowski opiera się na założeniu niezależności, w ramach danej klasy, wartości poszczególnych cech od siebie. Prowadzi to do oszacowania:

$$P(x_1, x_2, \ldots, x_r \mid c) \approx \prod_{k=1}^{r} P(x_k \mid c) \tag{2.1}$$

[1]Chodzi o przypadek, kiedy w jednym zbiorze występują takie same wektory, ale przypisane do różnych klas.

Wektor zostanie w efekcie zaliczony do klasy:

$$c_{NB} = \arg\max_{c \in C} P(c) \prod_{k=1}^{r} P(x_k \mid c) \tag{2.2}$$

Teraz należy oszacować jedynie czynniki $P(x_k \mid c)$. Możemy to zrobić biorąc pod uwagę wektory z c, dla których jedynie wartości cechy k są zbliżone do wartości x_k wektora klasyfikowanego. W oparciu o zbiór treningowy da się teraz badać wektory mniej podobne do tych, które on zawiera (konkretne metody opisane są w podrozdziale 2.2). Jest ich typowo o wiele więcej niż wynosi liczebność tego zbioru. W ten sposób system zyskał zdolność generalizacji, to znaczy potrafi sklasyfikować punkty nie należące do zbioru treningowego.

Warto zauważyć, że założenie o wzajemnej niezależności cech jest często nieprawdziwe. Jednak nawet jeżeli tak jest, naiwny klasyfikator Bayesowski potrafi osiągać dobre rezultaty. Dobrym przykładem jest klasyfikacja tekstu. Mitchell[1] opisuje przykład skutecznego wykorzystania tego algorytmu do oceny artykułów pod kątem ich przynależności do konkretnych grup dyskusyjnych.

2.2 Szacowanie prawdopodobieństw

Przedstawianie sposobów szacowania prawdopodobieństw występujących we wzorze (2.2) warto rozpocząć od prawdopodobieństw klas („a priori"). Mogą być one estymowane wzorem:

$$P(c) = \frac{n_c}{n} \tag{2.3}$$

Jest to metoda analogiczna do prostego zliczania, opisanego w podrozdziale 2.2.1. Zamiast niej można zastosować inne algorytmy zawarte w 2.2.1, ale w przypadku prawdopodobieństw „a priori" nie są one szczególnie użyteczne.

Prawdopodobieństwa warunkowe ze wzoru (2.2) wymagają bardziej szczegółowego omówienia. Ze względu na zupełnie odmienny sposób wyznaczania osobno opisane zostaną metody dotyczące prawdopodobieństw warunkowych dla cech o wartościach symbolicznych (nieuporządkowanych) i numerycznych (uporządkowanych).

2.2.1 Cechy symboliczne

Proste zliczanie

Najprostsza metoda wyznaczania $P(x_k \mid c)$ dla cech symbolicznych nosi nazwę prostego zliczania. Zliczane są punkty należące do c, dla których cecha numer k przyjmuje wartość taką samą, jak w wektorze klasyfikowanym. Prawdopodobieństwo jest przybliżane przez stosunek ich liczby do liczby wszystkich wektorów należących do klasy c:

$$P(x_k \mid c) = \frac{n_{x_k \wedge c}}{n_c} \tag{2.4}$$

Rozpatrzmy przypadek, kiedy w klasie c nie znajdzie się żaden wektor, dla którego $x_k^{(j)} = x_k$. Prawdopodobieństwo $P(x_k \mid c)$ wyniesie wtedy zero i w efekcie klasa c zostanie definitywnie wykluczona z dalszych poszukiwań, ponieważ iloczyn we wzorze (2.2) zawiera zero. Zdarzenie takie jest szczególnie prawdopodobne, jeżeli $P(x_k \mid c)$ jest rzeczywiście niewielkie, a zbiór treningowy nieliczny. W rzeczywistości wszystkie prawdopodobieństwa szacowane dla mało licznych klas są mało wiarygodne.

Metoda no-match

Rozwiązanie problemu zerowych zliczeń przynosi metoda no-match. Podstawą jej działania jest proste zliczanie. Zerowe wartości $n_{x_k \wedge c}$ w liczniku są jednak zastępowane przez mały czynnik dodatni, aby uniknąć zerowania całego iloczynu. W [2] zaproponowano wartość tego czynnika równą $P(c)/n$, co daje wzór:

$$P(x_k \mid c) = \frac{P(c)/n}{n_c} = \frac{n_c}{n^2 n_c} = \frac{1}{n^2} \tag{2.5}$$

Korekta Laplace'a (ang. Laplace correction)

Innym podejściem do problemu dokładności szacowanych prawdopodobieństw dla mało licznych klas jest korekta Laplace'a. Modyfikuje ona jednakowo wszystkie prawdopodobieństwa zgodnie ze wzorem:

$$P(x_k \mid c) = \frac{n_{x_k \wedge c} + f}{n_c + f l_k}, \tag{2.6}$$

gdzie f to stała. Testy opisane w [2] pokazują, że najlepszych wyników można się spodziewać dla $f = 0,01$, a nieco gorszych dla $f = 1$, która to wartość jest powszechnie stosowana.

m-oszacowanie (ang. m-estimate)

Podobnie działa metoda m-oszacowania. Jeśli rozkład wartości cechy k oznaczymy jako p_k, a m będzie arbitralnie dobraną stałą, otrzymamy:

$$P(x_k \mid c) = \frac{n_{x_k \wedge c} + mp_k(x_k)}{n_c + m} \tag{2.7}$$

Typowo, gdy nie są dostępne dodatkowe informacje o danych, zakładamy jednostajny rozkład prawdopodobieństwa, to znaczy $p_k = 1/l_k$ dla wszystkich wartości przyjmowanych przez atrybut numer k.

Warto zwrócić uwagę na jeszcze jeden aspekt używania m-oszacowania. Otóż modyfikuje ono wszystkie prawdopodobieństwa przybliżając ich wartość do p_k. Dla zbyt dużych m człon dodany zaczyna dominować nad zliczeniami dotyczącymi rzeczywistego zbioru treningowego. Powstaje w ten sposób model, w którym wszystkie klasy są jednakowo prawdopodobne.

2.2.2 Cechy uporządkowane

Cechę określamy jako uporządkowaną, kiedy przyjmowane przez nią wartości mają dobrze określoną kolejność oraz odległości pomiędzy sobą. Najczęściej są one rzeczywiste lub całkowite, chociaż można sobie wyobrazić ustawienie elementów symbolicznych w ciąg i przypisanie im np. kolejnych liczb naturalnych (o ile taka operacja ma akurat jakiś sens). Do cech uporządkowanych stosuje się metody znane z analizy danych ciągłych.

Rozkład normalny

Tradycyjnym podejściem do szacowania prawdopodobieństw dla atrybutów o wartościach uporządkowanych jest założenie, że podlegają one rozkładowi normalnemu. W takim wypadku zamiast $P(x_k \mid c)$ używamy wartości funkcji Gaussa:

$$P(x_k \mid c) \leftarrow g(x_k, \mu_c, \sigma_c) = \frac{1}{\sqrt{2\pi}\sigma_c} e^{-\frac{(x_k - \mu_c)^2}{2\sigma_c^2}}, \tag{2.8}$$

gdzie μ_c jest średnią wartością cechy k dla wektorów należących do c, natomiast σ_c — odchyleniem standardowym tej średniej.

Oszacowanie jądrowe (ang. kernel density estimation)

Inną ciekawą metodę wyznaczania prawdopodobieństw $P(x_k \mid c)$ znaleźć można w [3]. Funkcja rozkładu wartości cechy jest w niej przybliżana nie przez pojedynczy rozkład normalny, ale przez tzw. oszacowanie jądrowe (kernel density estimation). Jest ono sumą funkcji Gaussa:

$$P(x_k \mid c) = \frac{1}{n_c} \sum_{j:x^{(j)} \in c} g(x_k, x_k^{(j)}, \sigma_c) \tag{2.9}$$

$g()$ jest funkcją Gaussa określoną jak w równaniu (2.8). Konwencja przyjęta w [3] określa $\sigma_c = \sqrt{n_c}$.

2.3 Złożoność obliczeniowa

Spośród metod szacowania prawdopodobieństw opisanych w podrozdziale 2.2 można wyodrębnić trzy grupy metod różniących się złożonością obliczeniową algorytmów oraz ilością potrzebnej pamięci. Są one opisane w kolejnych podrozdziałach.

Na potrzeby omawianych tu zagadnień wprowadzić należy dodatkowe oznaczenia. Niech U oznacza liczbę cech o wartościach uporządkowanych, a N — nieuporządkowanych. K będzie oznaczać ilość klas, a W i W_t — odpowiednio liczbę wektorów zbioru treningowego i testowego. Średnią liczbę różnych wartości dla wszystkich atrybutów symbolicznych będziemy zapisywać jako S.

2.3.1 Metody działające na danych symbolicznych

Proces uczenia w przypadku powyższych metod polega na wyznaczeniu wartości $n_{x_k \wedge c}$ (patrz wzory 2.4 do 2.7). W tym celu należy sprawdzić wartości atrybutów symbolicznych w każdym wektorze zbioru treningowego. Operacja ta ma złożoność rzędu $W\ N$. Wyznaczonych wartości $n_{x_k \wedge c}$ jest KNS i tyle właśnie jednostek pamięci potrzeba do przechowywania wyników uczenia.

Klasyfikacja jednego punktu wymaga jednej operacji dzielenia (wzory 2.4 do 2.7) dla każdej cechy nieuporządkowanej. Zatem klasyfikacja W_t wektorów będzie procesem rzędu $W_t N$.

2.3.2 Przybliżanie rozkładem normalnym

Proces uczenia polega, w tym przypadku, na wyznaczeniu i zapamiętaniu wartości średnich ramach każdej klasy dla wszystkich cech uporządkowanych i odchyleń standardowych tych średnich również wewnątrz klas.

Wyznaczanie średnich w ramach klas dla jednej cechy jest operacją rzędu W, podobnie jak wyznaczanie odchyleń standardowych. W pierwszym przypadku wykonywanych jest W dodawań (wartość danej cechy jest dodawana do licznika odpowiadającego klasie, do jakiej należy dany wektor), po których następuje K dzieleń. Ponieważ zazwyczaj zachodzi $W > K$, odpowiada to złożoności rzędu W. Wyznaczanie odchyleń standardowych jest kilkukrotnie (stała) kosztowniejsze, ponieważ dla każdego wektora należy obliczyć kwadrat odchylenia od średniej dla danej klasy. Ma zatem tą samą złożoność.

Serie (dla wszystkich klas) średnich i odchyleń wyznaczane są dla wszystkich atrybutów uporządkowanych, a więc U razy. Daje to razem złożoność rzędu WU. Do zapisania wyznaczonych współczynników potrzeba $2KU$ jednostek pamięci.

Klasyfikacja sprowadza się do jednokrotnego wyznaczenia wartości funkcji Gaussa dla każdej cechy uporządkowanej w każdym nowym wektorze. Zatem klasyfikacja W_t punktów ma złożoność rzędu W_tU.

2.3.3 Oszacowanie jądrowe

Klasyfikacja przy użyciu oszacowania jądrowego nie wymaga wcześniejszego uczenia. Podczas rozpatrywania nowego przypadku, przy szacowaniu $P(x_k \mid c)$ są brane pod uwagę wszystkie wektory należące do klasy c. Ponieważ ilość wyrażeń $P(x_k \mid c)$, które trzeba oszacować, wynosi KU, klasyfikacja jednego wektora jest rzędu $W\ U$, a W_t wektorów — $W_tW\ U$. Dodatkowo w czasie klasyfikacji potrzebna jest pomocnicza pamięć o wielkości K.

2.3.4 Podsumowanie

Złożoności wszystkich algorytmów zestawione zostały w tabeli 2.1, natomiast porównanie ilości potrzebnej pamięci znajduje się w tabeli 2.2. W ostatniej kolumnie tabeli 2.1 znajdują się oszacowania dotyczące 10-krotnej kroswalidacji. Oznacza ona podział zbioru treningowego na 10 części i używanie każdej z nich po kolei jako zbioru testowego, a reszty — do procesu uczenia.

Metody	(A)	(B)	(C)	(D)
symboliczne	$W\ N$	W_tN	$(W+W_t)N$	WN
rozkład normalny	WU	W_tU	$(W+W_t)U$	WU
oszacowanie jądrowe	0	$W\ W_tU$	$W\ W_tU$	W^2U

Tabela 2.1: Złożoność obliczeniowa: (A) procesu uczenia na zbiorze W punktów; (B) klasyfikacji W_t punktów; (C) łącznie (A) i (B); (D) 10-krotnej kroswalidacji wewnątrz zbioru W punktów;

Metody	Proces uczenia	Klasyfikacja 1 wektora
symboliczne	KNS	0
rozkład normalny	$2KU$	0
oszacowanie jądrowe	0	K

Tabela 2.2: Pamięć potrzebna do zapisania wyników procesu uczenia i pomocnicza przy klasyfikacji nowego przypadku.

Jest to równoznaczne z 10 krotnym uczeniem na $9/10W$ wektorach i testowaniem na $1/10W$ wektorach.

Dodać należy, że oprócz powyższych kosztów model w procesie uczenia wykorzystuje dodatkowo K jednostek pamięci i wykonuje tyle samo operacji celem wyznaczenia i zapamiętania prawdopodobieństw „a priori" (patrz równanie 2.3). Również podczas klasyfikacji wartości prawdopodobieństw $P(c \mid x_1, x_2, \ldots, x_r)$ są przechowywane w tablicy o wielkości K. Klasyfikacja polega na wybraniu maksimum z tych wartości, co jest operacją rzędu K. Złożoność obu tych czynności jest zaniedbywalna w porównaniu z kosztami działania poszczególnych metod (patrz kolumna (C) w tabeli 2.1).

Porównując złożoności wszystkich algorytmów można zauważyć, że koszt oszacowania jądrowego jest znacząco wyższy niż pozostałych metod. Jego złożoność zależy od iloczynu liczebności zbiorów: treningowego i testowego, podczas kiedy na przykład przybliżanie rozkładem Gaussa — od ich sumy. W typowej sytuacji, kiedy wielkości te mają zbliżoną wartość, odpowiada to zależności odpowiednio: kwadratowej i liniowej, co zostało uwidocznione na przykładzie złożoności 10-krotnej kroswalidacji (kolumna (D) tabeli 2.1). Jest to istotna różnica, która znacznie utrudnia stosowanie oszacowania jądrowego do dużych zbiorów danych.

2.4 Ranking prawdopodobieństw

Warto podkreślić, że chociaż głównym zadaniem naiwnego klasyfikatora Bayesowskiego jest znalezienie najbardziej prawdopodobnej klasy dla testowanego wektora, jego możliwości są nieco większe. Przyjrzyjmy się wielkościom we wzorze (2.2), spośród których szukamy maksimum ze względu na c. W rzeczywistości są one proporcjonalne do prawdopodobieństw przynależności wektora do poszczególnych klas. Aby otrzymać prawdopodobieństwa $P(c \mid x_1, x_2, \ldots, x_r)$, wystarczy znormalizować je tak, aby dawały w sumie 1. W ten sposób otrzymujemy coś więcej niż ranking klas. Możemy sprawdzić, na ile pewna jest otrzymana klasyfikacja, czyli o ile pierwsza (w rankingu) klasa jest bardziej prawdopodobna od drugiej w kolejności. Żadna klasa nie jest wykluczona z rankingu. Daje to możliwość sprawdzenia prawdopodobieństwa nawet mało prawdopodobnych klas, co może być interesujące w pewnych zastosowaniach.

Rozdział 3

Implementacja naiwnego klasyfikatora Bayesowskiego

3.1 Modele w *GhostMiner*

W skład systemu *GhostMiner* wchodzą dwa moduły: *GhostMiner Developer* i *GhostMiner Analyzer*. Pierwszy służy do budowania modeli używanych później do analizy danych. Jest ona realizowana przy pomocy drugiego ze wspomnianych modułów.

Praca w programie *GhostMiner Developer* odbywa się w ramach projektów. Każdy projekt jest związany z konkretnym zbiorem danych. Wszystkie informacje o projekcie zawiera drzewo projektu. Jego korzeniem jest opis zbioru danych. W węzłach leżących poniżej mogą się znajdować dostępne w systemie transformatory danych, klasteryzatory, wizualizatory, klasyfikatory i narzędzia do ich testowania (krosswalidacja i X-test). Podstawowa zasada jest w każdym przypadku taka sama. Model składowy dostaje na wejściu dane z wyższego węzła drzewa, a rezultaty jego działania mogą być dostępne dla modeli znajdujących się w niżej położonych węzłach.

Dodanie do drzewa projektu nowego klasyfikatora przebiega następująco. Najpierw należy wybrać węzeł, do którego ten klasyfikator ma być przypisany. Po wyborze rodzaju klasyfikatora użytkownik ma możliwość jego konfiguracji. Następnie ma miejsce proces uczenia.

Użytkownik ma możliwość ponownej konfiguracji klasyfikatora, po której powinien nastąpić powtórny proces uczenia.

Aby zapewnić poprawne wykonywanie opisanych powyżej operacji każdy

klasyfikator będący częścią programu *GhostMiner* ma ściśle określoną strukturę. Do konfiguracji procesu uczenia służy *panel konfiguracyjny*. Uzyskane w ten sposób ustawienia są zapisywane jako *parametry*. Są one następnie przekazywane do części realizującej algorytmy uczenia i klasyfikacji, którą nazywać będziemy *modelem*.

Te trzy podstawowe części zostały zaimplementowane jako oddzielne klasy dziedziczące po klasach bazowych *GhostMiner'a*. Klasa realizująca *panel konfiguracyjny* dziedziczy po *TNBayesVis*, *parametry* są zaimplementowane w klasie dziedziczącej po *GMParameters*, a klasa *modelu* dziedziczy po *ClassifierBase*. W przypadku opisywanej w tej pracy implementacji naiwnego klasyfikatora Bayesowskiego klasy te noszą następujące nazwy:

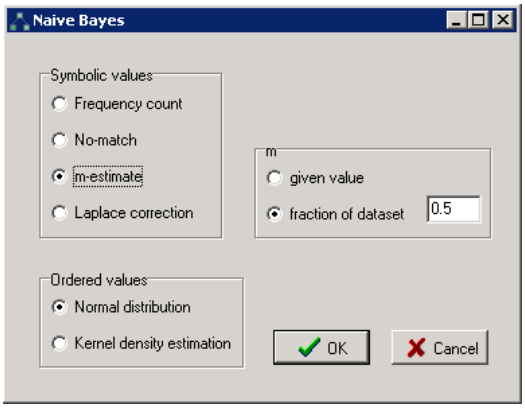
- panel konfiguracyjny: *TNBayesVis*;
- parametry: *NBayesP*;
- model: *NBayesM*;

Za każdym razem, gdy włącza się *panel konfiguracyjny*, odpowiednie jego pola są wypełniane wartościami aktualnych *parametrów*[1]. Dzieje się tak nawet wtedy, gdy klasyfikator jest po raz pierwszy dodawany do drzewa projektu. Domyślne wartości *parametrów* są w takim przypadku wcześniej ustawione, co zapewnia funkcja *NBayesP::Default*. Po zaakceptowaniu przez użytkownika zmian w konfiguracji następuje aktualizacja *parametrów* dokonywana przez funkcję *TNBayesVis::GetParameters*.

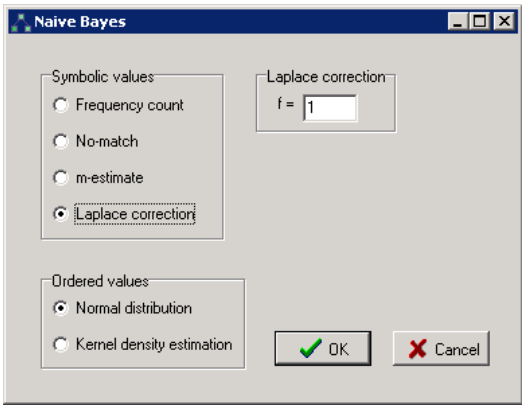
Przed rozpoczęciem procesu uczenia ustawienia zawarte w *parametrach* są przekazywane do *modelu*. Funkcja *NBayesM::SetParameters* wykonuje to zadanie przypisując wartości aktualnych *parametrów* ich lokalnej kopii. Korzysta w tym celu z przeciążonego operatora *NBayesP::=*.

Najważniejszymi metodami *modelu* są *NBayesM::RunModel* oraz *NBayesM::TestVector*. Pierwsza realizuje proces uczenia modelu. Druga służy do klasyfikacji wektorów. Jej argumentem jest wektor danych (*DataVector*). Zwraca ona numer klasy, do której zostaje on przyporządkowany. Dodatkowo funkcja ta może wpisać do tablicy, będącej drugim argumentem, prawdopodobieństwa przynależności punktu do każdej z klas.

[1]Zadanie to wypełnia funkcja *TNBayesVis::SetParameters*.



Rysunek 3.1: Panel konfiguracyjny: opcje dla m-oszacowania



Rysunek 3.2: Panel konfiguracyjny: opcje dla korekty Laplace'a

3.2 Szczegóły implementacji

3.2.1 TNBayesVis

Na panelu konfiguracyjnym (rysunki 3.1 i 3.2) znajdują się dwie grupy przycisków radiowych służących do wyboru metod liczenia prawdopodobieństw dla cech o wartościach symbolicznych i uporządkowanych. Dodatkowo dla korekty Laplace'a (rys. 3.2) i m-oszacowania (rys. 3.1) można wprowadzić wartości f i m (patrz równania 2.6 i 2.7). Drugi parametr może zostać podany bezpośrednio, bądź jako ułamek liczebności zbioru treningowego.

Metoda *ConfigureCheck* sprawdza poprawność wprowadzonych parame-

trów konfiguracyjnych. W wypadku wprowadzenia przez użytkownika błędnych lub sprzecznych wartości otrzyma on komunikat o błędzie zdefiniowany w tej funkcji. W naiwnym klasyfikatorze Bayesowskim istnieją zabezpieczenia przed wprowadzaniem jako m (na oba sposoby) i f liczb ujemnych.

3.2.2 NBayesP

Parametrami naiwnego klasyfikatora Bayesowskiego są:

- *OneGauss*: parametr o wartościach logicznych wskazujący, czy dla cech uporządkowanych ma zostać użyty rozkład Gaussa; jeśli nie, zastosowane zostanie oszacowanie jądrowe;
- *UnorderedChoice*: określa wybór algorytmu działającego na danych nieuporządkowanych; wartości całkowite;
- *GivenValue*: wielkość logiczna określająca, czy wartość m (patrz równanie 2.7) została podana bezpośrednio;
- m: liczba całkowita będąca parametrem m-oszacowania (wzór 2.7);
- *frac*: liczba rzeczywista określająca, jakim ułamkiem liczby wektorów treningowych ma być m (równanie 2.7);
- f: liczba rzeczywista występująca w korekcie Laplace'a (równanie 2.6);

3.2.3 NBayesM

RunModel

Prawidłowa inicjalizacja modelu przed rozpoczęciem procesu uczenia jest zapewniona przez wywołanie funkcji *Reset*. Usuwa ona tablice dynamicznie alokowane w czasie ewentualnego wcześniejszego uczenia i zeruje odpowiednie pola klasy. Następnie *RunModel* wykonuje trzy zadania:

- Prawdopodobieństwa klas ($P(c)$ ze wzoru (2.2)) są liczone i zapisywane w tablicy *ClassProbability*. W tym celu wykorzystywane są metody *GetClassCount* i *GetNrVectors* klasy *DataSet*. Zwracają one odpowiednio: liczebność określonej klasy i liczebność zbioru treningowego.

- Funkcja *Count* realizuje proces uczenia dla atrybutów nieuporządkowanych. Dla każdej takiej cechy zlicza ona wystąpienia wszystkich przyjmowanych przez nią wartości oddzielnie w ramach każdej klasy. Zliczenia są zapisywane w tablicy *UCounter* zgodnie ze schematem: *UCounter[cecha][klasa][wartość]=liczba wystąpień*.

- Jeśli model został skonfigurowany do przybliżania rozkładu cech uporządkowanych rozkładem Gaussa, wywoływana jest funkcja *LearnGaussian*. Alokuje ona tablice *GaussMean* oraz *GaussStd*. Zapisuje do nich wartości średnie i ich odchylenia standardowe we wszystkich klasach dla cech uporządkowanych. Dodatkowo najmniejsze niezerowe odchylenia standardowe średnich w ramach danej cechy są zapamiętywane w tablicy *StdMin*.

TestVector

TestVector rozpoczyna działanie od skopiowania prawdopodobieństw klas z *ClassProbability* do roboczej tablicy *Prob*. Następnie dla każdej cechy o znanej wartości szacowane są odpowiednie $P(x_k \mid c)$, przez które na bieżąco wymnażane są odpowiadające im pola *Prob*. Zadanie to wypełnia grupa funkcji, o nazwach rozpoczynających się przedrostkiem *Test...* (określanych dalej jako *TestXXX*). Implementują one algorytmy opisane w rozdziałach 2.2.1 i 2.2.2. Jako argumenty otrzymują tablicę *Prob*, którą modyfikują, oraz testowany wektor i numer rozpatrywanej aktualnie cechy.

W zależności od typu atrybutu i wybranych przez użytkownika ustawień modelu mogą zostać użyte następujące funkcje *TestXXX*:

- dla cech nieuporządkowanych:

  - *TestFreq* implementuje algorytm prostego zliczania (patrz równanie 2.4);
  - *TestNoM* realizuje metodę no-match (wzór 2.5);
  - *TestMEst*: m-oszacowanie (wzór 2.7);
  - *TestLap*: korekta Laplace'a (wzór 2.6);

- dla cech uporządkowanych:

– *TestGauss* wymnaża wszystkie pola *Prob* przez wartości funkcji Gaussa zgodnie ze wzorem (2.8). Korzysta w tym celu z funkcji *Gauss*[2], której parametry określone są w tablicach *GaussMean* i *GaussStd*. Dodatkowo za każdym razem wartość funkcji Gaussa jest mnożona przez właściwą danej cesze wartość *StdMin*, czyli najmniejszego niezerowego odchylenia standardowego średniej w ramach danej cechy. Zabieg ten chroni przed wyjściem poza zakres używanego typu rzeczywistego. Występowanie takich zdarzeń zostało stwierdzone dla danych z dużą ilością cech o stosunkowo małym odchyleniu standardowym. Za przykład służyć może zbiór *sonar*. Składa się on z wektorów o 60 cechach uporządkowanych, których odchylenie standardowe waha się pomiędzy $0,26$ i $0,0045$. Dla uproszczenia obliczeń przyjmijmy dalej, że odchylenia te są równe i wynoszą $0,03$. Niech wartości cech klasyfikowanego wektora będą odległe od średnich o $0,05$. W takim przypadku wartości zwracane przez funkcję *Gauss* będą większe niż 26. Po wymnożeniu 60 takich wartości otrzymamy liczbę większą od $3,8 \times 10^{85}$, co poskutkuje przekroczeniem zakresu. Dzięki zastosowanym środkom zapobiegawczym maksimum rozkładu o najmniejszym odchyleniu będzie równe 1.

– *TestKernel* realizuje metodę oszacowania jądrowego. Składniki sumy występującej w równaniu (2.9) są w pętli po wszystkich wektorach zbioru treningowego dosumowywane do odpowiednich pól pomocniczej tablicy p (pola te zawierają prawdopodobieństwa $P(x_k \mid c)$). Następnie przy ich pomocy zmieniane są pola tablicy *Prob*.

[2]Funkcja ta zwraca wartość rozkładu Gaussa z pominięciem stałej normującej $1/\sqrt{2\pi}$, jednakowej dla wszystkich klas.

Rozdział 4

Wyniki testowe

Naiwny klasyfikator Bayesowski został przetestowany na jedenastu zbiorach danych. Zbiory *wine, pima-indian-diabetes, ionosphere, vowel* i *iris* zawierają wektory o cechach uporządkowanych, a *vote, splice, kr-vs-kp, mushroom* i *soybean-large* — o cechach nieuporządkowanych. Wektory należące do zbioru *breast-cancer-wisconsin* mają atrybuty o wartościach całkowitych od 1 do 10, i dlatego został on użyty zarówno do testowania metod działających na danych symbolicznych, jak i uporządkowanych. Podstawowe wielkości opisujące użyte zbiory danych zostały przedstawione w tabeli 4.1.

Wyniki przedstawione w tabelach 4.2 do 4.8 są rezultatem 10-krotnego powtórzenia 10-krotnej kroswalidacji. Zostały one zapisane według schematu: $X/Y/Z$. X jest w tym przypadku średnią dokładnością, Y — średnim odchyleniem wewnętrznych wyników kroswalidacji, a Z — odchyleniem średniej dokładności (X).

Przez n oznaczono tutaj, podobnie we wcześniejszej części pracy, liczebność zbioru treningowego.

Tabela 4.9 zawiera podsumowanie wyników osiąganych przez poszczególne metody szacowania prawdopodobieństw; jest to porównanie ich dokładności parami biorąc pod uwagę wszystkie zbiory danych. Do sprawdzenia, czy na danym zbiorze jedna metoda okazała się dokładniejsza od innej, użyto rozkładu t-Studenta dla 9 stopni swobody (ponieważ porównywano wyniki 10-krotnej kroswalidacji) z poziomem ufności $\alpha = 0,95$. Sytuację, w której przewagi żadnego z algorytmów nie można było określić z zadanym poziomem ufności, zaliczano do kategorii „wyników podobnych". Następnie uzyskane wyniki zostały zsumowane dla wszystkich zbiorów. Rezultaty przedstawione w tej tabeli podzielone są na trzy części.

Zbiór danych	Ilość wektorów	Ilość klas	Ilość cech	Nieznane wartości	Najliczniejsza klasa [%]
Wartości nieuporządkowane					
vote	435	2	16	392	61,38
splice	3190	3	60	0	51,88
kr-vs-kp	3196	2	36	0	52,22
mushroom	8124	2	22	2480	51,8
soybean-large	290	15	35	390	13,79
breast-cancer-wisconsin	699	2	9	16	65,52
Wartości uporządkowane					
wine	178	3	13	0	39,89
pima-indians-diabetes	768	2	8	0	65,1
ionosphere	351	2	34	0	64,1
vowel	871	6	7	0	19,75
iris	150	3	4	0	33,33
breast-cancer-wisconsin	699	2	9	16	65,52

Tabela 4.1: Zbiory danych użyte do testowania dokładności naiwnego klasyfikatora Bayesowskiego.

Zbiór	Dokładność [%]	
	Rozkład normalny	Oszacowanie jądrowe
wine	97,71 / 3,71 / 0,60	**99,10** / 2,18 / 0,50
pima-indians-diabetes	**75,62** / 5,24 / 0,34	67,07 / 2,38 / 0,14
ionosphere	86,50 / 4,63 / 0,30	**91,34** / 5,17 / 0,34
vowel	79,16 / 3,97 / 0,16	**82,24** / 3,88 / 0,40
iris	**95,40** / 5,08 / 0,40	95,00 / 5,71 / 0,73
breast-cancer-wisconsin	**96,01** / 2,32 / 0,09	94,79 / 2,24 / 0,22

Tabela 4.2: Dokładność naiwnego klasyfikatora Bayesowskiego; wartości uporządkowane.

Metoda	Dokładność [%]
proste zliczanie	90,21 / 4,81 / 0,16
no-match	90,25 / 3,94 / 0,12
korekta Laplace'a ($f = 1$)	90,11 / 4,46 / 0,25
korekta Laplace'a ($f = 0,01$)	**90,28** / 4,70 / 0,16
m-oszacowanie ($m = 0,1n$)	90,05 / 5,07 / 0,17
m-oszacowanie ($m = 0,01n$)	90,00 / 4,46 / 0,16

Tabela 4.3: Dokładność naiwnego Bayesa na zbiorze *vote*; wartości nieuporządkowane.

Metoda	Dokładność [%]
proste zliczanie	95,48 / 1,23 / 0,07
no-match	95,55 / 1,17 / 0,09
korekta Laplace'a ($f = 1$)	95,40 / 1,29 / 0,12
korekta Laplace'a ($f = 0,01$)	**95,57** / 1,03 / 0,10
m-oszacowanie ($m = 0,1n$)	94,01 / 1,41 / 0,07
m-oszacowanie ($m = 0,01n$)	94,98 / 1,16 / 0,14

Tabela 4.4: Dokładność naiwnego Bayesa na zbiorze *splice*; wartości nieuporządkowane.

Metoda	Dokładność [%]
proste zliczanie	**87,88** / 2,25 / 0,15
no-match	87,83 / 1,89 / 0,14
korekta Laplace'a ($f = 1$)	87,81 / 1,83 / 0,16
korekta Laplace'a ($f = 0,01$)	87,84 / 1,64 / 0,15
m-oszacowanie ($m = 0,1n$)	83,78 / 2,05 / 0,16
m-oszacowanie ($m = 0,01n$)	87,20 / 2,06 / 0,20

Tabela 4.5: Dokładność naiwnego Bayesa na zbiorze *kr-vs-kp*; wartości nieuporządkowane.

Metoda	Dokładność [%]
proste zliczanie	**99,68** / 0,23 / 0,03
no-match	99,60 / 0,23 / 0,03
korekta Laplace'a ($f = 1$)	95,48 / 0,72 / 0,02
korekta Laplace'a ($f = 0,01$)	99,09 / 0,35 / 0,02
m-oszacowanie ($m = 0,1n$)	92,17 / 0,85 / 0,02
m-oszacowanie ($m = 0,01n$)	93,56 / 0,85 / 0,03

Tabela 4.6: Dokładność naiwnego Bayesa na zbiorze *mushroom*; wartości nieuporządkowane.

Metoda	Dokładność [%]
proste zliczanie	82,31 / 6,67 / 0,40
no-match	**92,52** / 3,93 / 0,60
korekta Laplace'a ($f = 1$)	84,62 / 5,15 / 0,34
korekta Laplace'a ($f = 0,01$)	92,21 / 4,70 / 0,55
m-oszacowanie ($m = 0,1n$)	70,52 / 5,95 / 0,70
m-oszacowanie ($m = 0,01n$)	85,34 / 6,16 / 0,50

Tabela 4.7: Dokładność naiwnego Bayesa na zbiorze *soybean-large*; wartości nieuporządkowane.

Metoda	Dokładność [%]
proste zliczanie	96,62 / 2,33 / 0,16
no-match	97,37 / 2,12 / 0,08
korekta Laplace'a ($f = 1$)	97,37 / 1,93 / 0,08
korekta Laplace'a ($f = 0,01$)	**97,38** / 1,56 / 0,10
m-oszacowanie ($m = 0,1n$)	96,93 / 0,26 / 0,02
m-oszacowanie ($m = 0,01n$)	97,35 / 2,01 / 0,08

Tabela 4.8: Dokładność naiwnego Bayesa na zbiorze *breast-cancer-wisconsin*; wartości nieuporządkowane.

Porównywane metody	Lepsza pierwsza	Lepsza druga	Podobne wyniki
korekta Laplace'a z $f = 1$ i z $f = 0,01$	-	2	4
m-oszacowanie z $m = 0,1n$ i z $m = 0,01n$	-	5	1
korekta Laplace'a i m-oszacowanie	4	-	2
korekta Laplace'a i proste zliczanie	2	1	3
m-oszacowanie i proste zliczanie	2	3	1
no-match i proste zliczanie	2	1	3
korekta Laplace'a i no-match	-	1	5
no-match i m-oszacowanie	4	-	2
rozkład normalny i oszacowanie jądrowe	2	2	2

Tabela 4.9: Porównanie dokładności metod naiwnego Bayesa na wszystkich zbiorach.

Pierwsza część tabeli 4.9 zawiera porównanie dokładności algorytmów korekty Laplace'a oraz m-oszacowania dla różnych wartości parametrów (odpowiednio f i m, patrz wzory 2.6 i 2.7). Wobec znaczących różnic w dokładności poszczególnych ustawień, w drugiej części tabeli użyto do porównań różnych metod wartości parametrów dających lepsze wyniki ($f = 0,01$ i $m = 0,01n$). Część druga jest zestawieniem (parami) czterech algorytmów działających na wartościach nieuporządkowanych, a część trzecia dotyczy metod dla wartości uporządkowanych.

Podsumowanie

Celem tej pracy była implementacja i opis modułu realizującego naiwny klasyfikator Bayesowski, działający jako część programu *GhostMiner*. W rozdziale 4 przedstawione zostały uzyskane w wyniku testów dokładności różnych algorytmów szacujących prawdopodobieństwa występujące w równaniu (2.2), będącym podstawą naiwnego Bayesa. Pozwalają one na wyciągnięcie kilku wniosków.

Porównując średnią dokładność klasyfikacji zarówno m-oszacowania jak i korekty Laplace'a dla różnych konfiguracji (patrz pierwsza część tabeli 4.9) można zauważyć, że obie metody osiągają lepsze wyniki dla mniejszych wartości parametrów m i f. Większej dokładności można się zatem spodziewać stosując takie parametry, które w niewielkim stopniu modyfikują szacowane prawdopodobieństwa.

Druga część tabeli 4.9 daje możliwość uszeregowania poszczególnych metod ze względu na osiąganą dokładność klasyfikacji. Pierwsze miejsce w takim zestawieniu przyznać należy metodzie no-match, która w bezpośrednim porównaniu przewyższyła każdą inną metodę. Podobne wyniki zaobserwować można dla korekty Laplace'a. Niższą dokładność w porównaniu z tymi dwoma metodami osiągnęło proste zliczanie. Poniżej oczekiwań ocenić należy wyniki m-oszacowania, które zajmuje ostatnie miejsce osiągając dokładność porównywalną jedynie z prostym zliczaniem. Można podejrzewać, że przyjęta wartość m, równa jednej setnej liczebności zbioru treningowego, w zbyt dużym stopniu modyfikuje szacowane prawdopodobieństwa[1].

Porównując wyniki osiągane przez metody przybliżania rozkładem normalnym i oszacowania jądrowego nie można wskazać dokładniejszej z nich. Różnice w dokładności dla poszczególnych zbiorów danych wynikają ze specyfiki tych zbiorów. Co za tym idzie oba algorytmy stanowią użyteczne narzędzia klasyfikacyjne i wzajemnie się uzupełniają.

[1] Zjawisko to zostało bliżej wyjaśnione w rozdziale 2.2.1, na stronie 9.

Bibliografia

[1] Tom M. Mitchell: *Machine Learning*, The McGraw-Hill (1997)

[2] Ron Kohlavi, Barry Becker, Dan Sommerfield: *Improving Simple Bayes*, Data Mining and Visualization Group

[3] George H. John, Pat Langley: *Estimating Continuous Distributions in Bayesian Classifiers*, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo (1995)