Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych
Informatyka

Metody Odkrywania Wiedzy

Klasyfikator bayesowski typu AODE

Nie-całkiem-naiwny klasyfikator bayesowski typu AODE
(averaged one-dependence estimators).
Porównania ze standardowym naiwnym klasyfikatorem
bayesowskim i innymi algorytmami klasyfikacji dostępnymi w R.

Dokumentacja końcowa

Wykonali:
Paweł Guz
Mateusz Kędrzyński

Prowadzący:
dr inż. Paweł Cichosz

Warszawa, 08 VI 2015

1 Interpretacja tematu projektu

1.1 Wprowadzenie

Naiwny Klasyfikator Bayesa nie jest trudny w implementacji i często używany przy zagadnieniach klasyfikacji. Skuteczność jego działania zależy od dokładności estymowanych prawdopodobieństw atrybutów, które bazują na wzajemnej niezależności, co często nie jest spełnione. Jego odmiany: LBR jak i TAN polepszają jego działanie w zamian za znaczny koszt obliczeniowy. AODE ma być algorytmem zyskujący podobne rezultaty, bez nadmiernego narzutu obliczeniowego.

1.2 Naiwny klasyfikator Bayesa

Przyjmijmy następujące oznaczenia:

$x = \langle x_1, x_2, ..., x_n \rangle \in X$: przykład podlegający klasyfikacji

$y \in \{c_1, c_2, ..., c_k\}$: klasy klasyfikacji

$\hat{}$: wielkość związana z danymi trenującymi

T: zbiór zmiennych trenujących

Ze wzoru Bayesa otrzymujemy:

$$\Pr(y|\mathbf{x}) = \frac{\Pr(y, \mathbf{x})}{\Pr(\mathbf{x})} \tag{1}$$

a stąd:

$$\arg\max_y(\Pr(y|\mathbf{x})) = \arg\max_y(\Pr(y, \mathbf{x})) \tag{2}$$

Zakładając, że dane trenujące stanowią reprezentatywną próbkę, częstość wystąpienia danego zdarzenia w próbce będzie z wystarczająco dobrą aproksymacją równe prawdopodobieństwu tego zdarzenia.

$$\Pr(y, \mathbf{x}) \approx \hat{\Pr}(y, \mathbf{x}) \tag{3}$$

Jednakże, zakładając, że liczba atrybutów jest dostatecznie duża, możliwość wystąpienia próbki $(y, \mathbf{x})$ będzie względnie mała. Jedną z metod obejścia tego ograniczenia, jest estymacja tego prawdopodobieństwa poprzez inne prawdopodobieństwa, które możemy z większą pewnością uzyskać z próbek. Z definicji prawdopodobieństwa warunkowego otrzymujemy:

$$\Pr(y, \mathbf{x}) = \Pr(y) \Pr(\mathbf{x}|y) \tag{4}$$

W przypadku względnie małej liczby klas oraz względnie dużej liczby próbek, wartość $\Pr(y)$ jesteśmy w stanie wystarczająco dokładnie estymować. Jednakże estymowanie wartości $\Pr(\mathbf{x}|y)$ wciąż będzie kłopotliwe.

Naiwny Klasyfikator Bayesa zakłada niezależność atrybutów, co daje:

$$\Pr(\mathbf{x}|y) = \prod_{\mathbf{i=1}}^{\mathbf{n}} \Pr(\mathbf{x_i}|y), \tag{5}$$

Opierając się na wyżej wymienionych zależnościach, wybieramy klasę dla danej próbki na podstawie:

$$\arg\max_y(\hat{\Pr}(y) \prod_{i=1}^{n} \hat{\Pr}(\mathbf{x_i}|y)) \tag{6}$$

1.3 Averaged One-Dependence Estimators (AODE)

Algorytmem pozwalającym na osłabienie założenia o niezależności atrybutów jest Averaged One-Dependence (AODE). Opiera się na wybraniu atrybutów zależnych, których wartość dla danej próbki x w zbiorze trenującym występuje przynajmniej przyjęte m razy.

Wykorzystując zależność:

$$\Pr(y, \mathbf{x}) = \Pr(y, x_i) \Pr(\mathbf{x}|\mathbf{y}, \mathbf{x_i}) \tag{7}$$

oraz sumując po wszystkich atrybutach występujących w danych testowych dla danej wartości odpowiednią liczbę razy otrzymujemy ($\geqslant m$), otrzymujemy:

$$\Pr(y, \mathbf{x}) = \frac{\sum_{i:1 \leqslant i \leqslant n \ \wedge \ m \leqslant F(x_i)} \Pr(y, x_i) \Pr(\mathbf{x}|y, x_i)}{|i : 1 \leqslant i \leqslant n \ \wedge \ m \leqslant F(x_i)|} \tag{8}$$

Ponieważ mianownik jest taki sam dla każdej klas, wybór odpowiedniej klasy sprowadza się do rozważenia następującego zagadnienia:

$$\arg\max_y \Big(\sum_{i:1 \leqslant i \leqslant n \ \wedge \ m \leqslant F(x_i)} \hat{\Pr}(y, x_i) \prod_{j=1}^{n} \hat{\Pr}(x_j|y, x_i) \Big) \tag{9}$$

Złożoność pamięciowa podczas treningu jak i klasyfikacji jest taka sama i sprowadza się do utrzymywania tablicy trójwymiarowej (albo pięciowymiarowej) (v - średnia liczba wartości atrybutu, k - liczba klas, n - liczba atrybutów, a t - liczba próbek trenujących): $\mathcal{O}(k(nv)^2)$, złożoność czasowa w trakcie treningu: $\mathcal{O}(tn^2)$, a w trakcie klasyfikacji: $\mathcal{O}(kn^2)$.

Oczekujemy lepszych rezultatów od algorytmu AODE ze względu na mniejszy nacisk na niezależność argumentów. Ponadto, zaletą AODE jest możliwość inkrementalnego nauczania. W przypadku dodatkowych danych należy jedynie zaktualizować tabelę. Parametr m jest także niezależny od modelu.

2 Struktura projektu

Na projekt składają się następujące foldery:

- *data* - użyte zbiory danych
- *doc* - dokumentacja projektu
- *src* - implementacja algorytmu + testy

3 Opis implementacji

Implementacja algorytmu znajduje się w pliku *src/aode.R*. Funkcja *aode* jako parametry przyjmuje: $formula$ - parametr opisujący atrybut decyzyjny oraz *data* - dane wykorzystywane do nauki modelu. Po wczytaniu dostępnych klas oraz atrybutów do zmiennych odpowiednio *cl* oraz *attrs*, inicjalizowane są trzy tablice.

- T_{kv} - zliczająca wystąpienie każdej wartości każdego z atrybutów.
- T_{kvc} - zawierająca prawodpodobieństwo wystąpenia klas dla wartości każdego atrybutu.

- T_{kvkvc} - zawierająca prawodpodobieństwo wystąpenia klas dla każdych dwóch par wartość-atrybut.

Następnie zachodzi faza trenowania:

1. Zliczanie wystąpień odpowiednich wartości we wszystkich tabelach na podstawie danych trenujących.
2. Obliczanie prawdopodobieństw z wykorzystaniem estymaty Laplace'a.

Ostatecznie tworzony jest model, składający się z listy atrybutów, klas oraz wyżej wymienionych tablic.

Do predykcji została użyta funkcja *prediction.aode*, która jako parametry pobiera wyuczony model, wartość m - będącą parametrem algorytmu AODE, oraz zbiór danych. Pierwszym krokiem jest na podstawie tablicy T_{kv} wyznaczenie istotnych atrybutów dla badanego przykładu: spełniających warunek $m \leqslant F(x_i)$, gdzie x_i to wartość dla i-tego atrybutu, a funkcja $F(x_i)$ określa liczbę wystąpień w danych trenujących. Następnie zgodnie ze wzorem:

$$\sum_{i:1 \leqslant i \leqslant n \ \wedge \ m \leqslant F(x_i)} \hat{\Pr}(y, x_i) \prod_{j=1}^{n} \hat{\Pr}(x_j | y, x_i) \tag{10}$$

obliczany jest znormalizowany wektor wartości dla każdej klasy y.

Przyjęto, że dane wejściowe, będą zdyskretyzowane oraz nie zawierały wartości pustych.

4 Testy poprawności implementacji

W celu udowodnienia poprawności implementacji algorytmu AODE użyto następującego zbioru danych (plik: *data/weather.csv*): Tablica 1.

Z uwagi na małą liczbę przykładów zbudowano model dla $m = 1$ i dla każdego rekordu z danych trenujących wyznaczono klasę decyzyjną ($src/verification.R$). Zgodność z danymi wejściowymi wskazuje, że zaimplementowany algorytm może mieć rację bytu: Tablica 2.

outlook	**temperature**	**humidity**	**wind**	**play**
sunny	hot	high	normal	no
sunny	hot	high	high	no
overcast	hot	high	normal	yes
rainy	mild	high	normal	yes
rainy	cold	normal	normal	yes
rainy	cold	normal	high	no
overcast	cold	normal	high	yes
sunny	mild	high	normal	no
sunny	cold	normal	normal	yes
rainy	mild	normal	normal	yes
sunny	mild	normal	high	yes
overcast	mild	high	high	yes
overcast	hot	normal	normal	yes
rainy	mild	high	high	no

Tablica 1: Dane weryfikujące

no	**yes**	**predict**	**real**
0.809	0.191	no	no
0.868	0.132	no	no
0.242	0.758	yes	yes
0.340	0.660	yes	yes
0.142	0.858	yes	yes
0.524	0.476	no	no
0.211	0.789	yes	yes
0.706	0.294	no	no
0.130	0.870	yes	yes
0.121	0.879	yes	yes
0.247	0.753	yes	yes
0.257	0.743	yes	yes
0.089	0.911	yes	yes
0.583	0.417	no	no

Tablica 2: Weryfikacja algorytmu AODE

5 Wykorzystane algorytmu porównawcze

5.1 Naiwny Klasyfikator Bayesa

Opis algorytmu znajduje się w punkcie 1.2. Zostały wykorzystane i porównane przez nas dwie implementacje:

- *naiveBayes*, znajduje się w pakiecie *e*1071.
- *naive.bayes*, znajduje się w pakiecie *bnlearn*.

5.2 Klasyfikator Tree-Augmented naive Bayes (TAN)

Jest to podejście oparte na naiwnym klasyfkatorze Bayesa. W celu osłabienia warunku niezależności atrybutów, konstruowana jest funkcja $p(x_i)$, która wyznacza atrybut zależny. Klasa docelowa jest wyznaczana na podstawie:

$$\arg\max_y(\hat{\Pr}(y)\prod_{i=1}^{n}\hat{\Pr}(x_i|y,p(x_i))) \tag{11}$$

Wykorzystana przez nas implementacja *tree.bayes* pochodzi z pakietu *bnlearn*.

5.3 Klasyfikator Lazy Bayesian Rules (LBR)

Przy podejściu LBR dla każdego przykładu x (klasyfikator tworzony jest w trakcie klasyfikacji) wyznaczany jest zbiór W wartości pewnych atrybutów (wykorzystując specjalizowane algorytmy zachłanne). Następnie zakłada się niezależność spośród pozostałych atrybutów (nie wchodzących w skład W) dla danego W oraz y.
Implementacja przez nas użyta to LBR z biblioteki $RWeka$.

5.4 Drzewo decyzyjne

Ponadto, użyliśmy dwóch algorytmów, gdzie model jest odmienny od klasyfikatora Bayesa. Pierwszym z nich są drzewa decyzyjne, implementacja użyta to $C4.5$ z biblioteki $RWeka$.

5.5 Maszyna wektorów nośnych

Użyta przez nas implementacja maszyny wektorów nośnych to *svm* z biblioteki *e*1071.

6 Testy i porównania z innymi algorytmami

Został przetestowane przez nas 3 zbiory danych (src/tests.R):

- weather - bardzo mały zbiór danych, wykorzystany do wstępnego przyjrzenia się algorytmom
- car - mały zbiór danych
- mushroom - średni zbiór danych

W przypadku większych zbiorów danych, istniały problemy z wystarczającymi zasobami używanych przez nas maszyn obliczeniowych.

Wszystkie użyte przez nas dane zawierają atrybuty dyskretne. W przypadku braku wartości atrybutu, potraktowaliśmy brak atrybutu jako kolejną wartość atrybutu (problem istniał w danych mushrooms).

Czas wykonania poszczególnych funkcji wyznaczono na podstawie funkcji:

$$system.time(replicate(N, function(args, ...)))$$

gdzie N było dobierane eksperymentalnie, a czas odpowiednio uśredniany. W celu uniezależnienia się od architektury komputera, czas jest porównywany względem czasu wykonania testu z wykorzystaniem naiwnego klasyfikatora Bayesa z biblioteki *bnlearn*. Średnio ta implementacja dawała najmniejsze wartości.

6.1 Weather

6.1.1 Charakterystyka zbioru

- Liczba atrybutów: 4
- Liczba klas: 2
- Liczba przykładów: 14
- Podział: modele były oceniane oraz testowane na tym samym zbiorze danych

6.1.2 Wyniki

	Accuracy	Error	Recall/Sensivity	Precision	Specifity	FMeasure	Time
AODE ($m < 5$)	1	0	1	1	1	1	5.06
AODE ($m = 5, 6$)	0.93	0.07	1	0.9	0.8	0.95	NA
NB (e1071)	0.93	0.07	1	0.9	0.8	0.95	0.77
NB (bnlearn)	0.93	0.07	1	0.9	0.8	0.95	1
TAN	1	0	1	1	1	1	1.3
LBR	0.93	0.07	0.9	1	1	0.95	15.1
C4.5	1	0	1	1	1	1	16.2

Tablica 3: Wyniki uzyskane dla zbioru Weather

W przypadku tak małej liczby danych, możemy wyciągnąć jedynie prymitywne hipotezy. Wszystkie algorytmy w miarę dokładnie zaklasyfikowały dane wejściowe, na których były nauczone, z dokładnością do możliwości przeuczenia modelu. Tak mały zbiór danych nie pozwalał na stworzenie satysfakcjonującego modelu SVM, dla każdego przykładu przewidywał jedną klasę decyzyjną.

6.2 Cars

6.2.1 Charakterystyka zbioru

- Liczba atrybutów: 6

- Liczba klas: 4
- Liczba przykładów: 1728
- Dane uczące: 864 przykładów (50% wszystkich przykładów wybranych losowo)
- Dane testowe: 864 przykładów (pozostałe przykłady)

6.2.2 Wyniki

	Accuracy	Error	Time
AODE ($m < 205$)	0.911	0.089	67
AODE ($m = 205$)	0.913	0.087	NA
AODE ($m = 210$)	0.907	0.093	NA
AODE ($m = 250$)	0.894	0.106	NA
AODE ($m = 280$)	0.889	0.111	NA
NB (e1071)	0.866	0.134	5
NB (bnlearn)	0.878	0.122	1
TAN	0.931	0.069	1
LBR	0.932	0.068	34.2
C4.5	0.896	0.104	3.35
SVM	0.213	0.787	3.2

Tablica 4: Wyniki uzyskane dla zbioru Cars

Zauważmy, że najlepsze rezultaty dały dwa algorytmy: kosztowny obliczeniowo LBR oraz TAN, który z drugiej strony był bardzo szybki. $AODE$, ten, który miał dawać podobne rezultaty mniejszym kosztem obliczeniowym niz wyzej wymienione algorytmy, daje nieco gorsze wyniki i to z większym czasem wykonania. Kolejny jest algorytm $C4.5$, a następnie podobne rezultaty dają obie implementacje Naiwnego Bayesa, z tym że implementacja *bnlearn* okazała się 5-krotnie szybsza. Fatalne wręcz rezultat zwrócił algorytm SVM, co może dalej świadczyć o niewystarczającej ilości danych do wytrenowania odpowiedniego modelu.
Wpływ parametru m na wynik jest zależny od rozkładu wartości poszczególnych atrybutów. W zbiorze danych *car* liczba wystąpień wartości atrybutów zawiera się w przedziale [205, 305] dlatego też atrybut m powinien być mniejszy niż górna granica tego przedziału. Widać tutaj pierwszy z problemów w doborze parametru m - zależy od ilości danych uczących. W przypadku inkrementalnego tworzenia modelu parametr ten wraz z douczaniem modelu, powinien być zmieniany. Innym podejściem, jest ustalenie jego wartości jako ułamek liczba przykładów trenujących.

6.3 Mushrooms

6.3.1 Charakterystyka zbioru

- Liczba atrybutów: 22
- Liczba klas: 7
- Liczba przykładów: 8124

- Dane uczące: 50% wszystkich przykładów wybranych losowo
- Dane testowe: pozostałe przykłady

6.3.2 Wyniki

	Accuracy	Error	Time
AODE ($m = 0$)	0.666	0.334	535
AODE ($m = 300$)	0.668	0.332	NA
AODE ($m = 600$)	0.665	0.335	NA
AODE ($m = 1500$)	0.662	0.338	NA
AODE ($m = 2200$)	0.638	0.362	NA
AODE ($m = 2800$)	0.639	0.361	NA
AODE ($m = 3300$)	0.638	0.362	NA
AODE ($m = 3500$)	0.637	0.363	NA
NB (e1071)	0.643	0.357	10.3
NB (bnlearn)	0.656	0.344	1
TAN	0.669	0.331	1.03
LBR	0.669	0.331	7022
C4.5	0.616	0.384	2.06
SVM	0.619	0.381	29.86

Tablica 5: Wyniki uzyskane dla zbioru Mushrooms

Zgodnie z oczekiwaniami uzyskane przez nas rezultaty prezentują się w następujący sposób: najlepiej wypadł LBR oraz TAN. Ten pierwszy okazał się niezwykle nieefektywny czasowo, natomiast znowu TAN dał bardzo dobre rezultaty i był bardzo szybki. Niewiele gorszy, a i znacząco szybszy od LBR okazał się AODE. Kolejne były implementacje naiwnego Bayesa, a potem SVM, który na znacznie większych danych poradził sobie bardzo dobrze. Na końcu było drzewo decyzyjne C4.5.
Dotychczasowa analiza parametru m pokazuje nam, że jego wartość powinna być względnie mała, służąca jedynie do odfiltrowania atrybutów, których wartość dla danego przykładu występuje względnie małą liczbę razy. Ponadto, zbyt duża wartość parametru m będzie znaczyła, że jako atrybuty znaczące będziemy brać jedynie te, których dana wartość występuje bardzo często, co nie musi nam dać znaczącej wiedzy. Jednym z argumentów przemawiającym za podwyższeniem wartości m jest możliwy mniejszy czas wykonania, aczkolwiek w trakcie testów nie zauważono dużej różnicy.

7 Wnioski

Najlepszym algorytmem klasyfikacji okazał się algorytm TAN. Dawał zarówno bardzo dobre wyniki, jaki i był efektywny czasowo. LBR bardzo dobrze radził sobie ze zbiorem małych danych, natomiast w przypadku dużych zbiorów danych czas wykonania był bardzo duży. To jest spowodowane dużym nakładem obliczeniowym przy klasyfikacji każdego przykładu. Niewiele gorszy od nich był $AODE$. Wartość parametru m nie powinna być wielka, ale na tyle duża, aby wyodrębnić tylko istotne atrybuty. Drzewo decyzyjne dawało nieco gorsze rezultaty od zaprezentowanych powyżej. Ciekawym przypadkiem okazał się SVM, który na mniejszych danych zupełnie sobie

nie radził, w przypadku większych zbiorów danych jego rezultaty były znacznie lepsze. Wydaje się, że jego użyteczność w porównaniu do innych algorytmów rosłaby wraz ze wzrostem zbiorów danych. Bardzo ciekawym zjawiskiem okazało się porównanie obydwu implementacji naiwnego bayesa. Obydwa dawały podobne rezultaty, a ta z biblioteki *nblearn* działała zdecydowanie szybciej.

Literatura

[1] Paweł Cichosz, *Materiały do wykładu z MOW*

[2] Geoffrey I. Webb Janice R. Boughton Zhihai Wang, *Not so naive Bayes: Aggregating one-dependence estimators*. School of Computer Science and Software Engineering

[3] Paweł Cichosz, *Data Mining Algorithms: Explained Using R*