MCP

Capturing Big Data by Satisfiability

Prezentacja pracy autorstwa:
- Miki Herman
- Gernot Salzer

Wstęp cz.1

- Analiza logiczna danych jest częścią uczenia maszynowego
- Podstawową celem pracy próba scharakteryzowania dużego zbioru danych poprzez formułę zdaniową

Wstęp cz.2

- Praca miała na celu stworzenie formuły logicznej, która dla każdej grupy obserwacji będzie prawdziwa oraz fałszywa dla reszty. Nie uwzględnia ona wszystkich atrybutów obserwacji, aby zoptymalizować, zmniejszyć jej rozmiar.
- Formuła uogólnia obserwację, a więc oznacza to, że może być spełniona przez obserwacje spoza rozpatrywanej grupy obserwacji, co stwarza możliwości użycia jej jako klasyfikatora.

Niezbędne pojęcia

- Wprowadźmy sobie podstawy algebry boolowskiej
- Literał – zmienna, mogąca przyjmować wartość true, false
- Klauzula – literały połączone alternatywą(spójnikiem lub)
- Formuła w koniunkcyjnej postaci normalnej(CNF) – to zbiór klauzul połączonych operatorem koniunkcji

Niezbędne pojęcia 2

- Klauzula Horna – klauzula, która posiada co najwyżej jeden pozytywny literał(bez negacji)
- Klauzula dual Horna – klauzula, która posiada co najwyżej jeden literał negatywny

Klauzula Horna: $(\neg p \vee \neg q \vee r)$

Dual Horn: $(\neg p \vee q \vee r)$

Niezbędne pojęcia 3

- Klauzula bijunkcyjna – klauzula posiadająca co najwyżej 2 literały
- Klauzula afiniczna – klauzula mająca postać równania liniowego x1+...xk=b, gdzie b należy do zbioru {0,1}, + oznacza operacją XOR, a x1 to zmienna

Niezbędne pojęcia 4

- Formuła Horna, dual horna, bijunkcyjna oraz afiniczna to formuły składające się z koniunkcji wymienionych wcześniej typów klauzul
- Wprowadźmy sobie również pojęcia wektora, który oznacza zbiór bitów np. a = [0,1,1,1,0]
- Domknięcie zbioru(closure) – jest to zbiór takich wartości, do których można dojść przez jakąś operację na zbiorze. Np. jeśli taką operacją byłoby dodawanie to jest zbiór liczb, które można utworzyć poprzez dodawanie do siebie różnych kombinacji liczb ze zbioru

Niezbędne pojęcia 5

- Domknięcia horna dla zbioru wektorów a oraz b, to taki, wektor, który powstaje w wyniku koniunkcji każdego elementu z a oraz b na odpowiednich pozycjach di = ai ^ bi
- Domknięcie dual horna – analogicznie do Horna tylko dokonuje się alternatywy każdego elementu
- Domknięcie bijunkcji – dla wektorów a, b oraz c, to taki wektor d, który na każdej pozycji przyjmuje 1 tylko gdy co najmniej dwie wartości z wektorów przyjmują 1 na tej samej pozycji
- Domknięcie afiniczne – wektor, który powstaje w wyniku wykonania operacji XOR na każdym elemencie wektorów a,b i c

Uogólnienie

- Dla danego zbioru S, który składa się z wektorów binarnych. Domknięcie C – będące domknięciem horna, dual horna, bijunkcji lub afinicznym jest zbiorem spełnień pewnej formuły logicznej H zbudowanej z operatorów domknięcia C, np. dla domknięcia horna będą to koniunkcja, alternatywa i negacja

Problem rozwiązywany przez system

- Problem multicharakteryzacji MCP:
- Dla danych dwóch zbiorów wektorów binarnych T,F o arności k należy znaleźć formułę Horna, dual horna, bijunkcyjną lub afiniczną, która jest spełniona dla wszystkich wektorów z T oraz jest niespełniona dla wszystkich wektorów z F
- Wykorzystanie tych 4 typów formuł nie jest przypadkowe, ponieważ sprawdzenia ich spełnialności jest możliwe w czasie wielomianowym.

PAC - Learning

- Dany problem jest instacją tzw. problemu PAC Learningu. Problem PAC-learning (Probably Approximately Correct learning) to problem uczenia maszynowego, w którym chcemy znaleźć algorytm, który na podstawie próbek (przykładów) danych uczących, będzie w stanie nauczyć się pewnej funkcji celu z jak najmniejszym prawdopodobieństwem popełnienia błędu.
- Problem PAC-learning jest ważnym zagadnieniem w uczeniu maszynowym, ponieważ wiele problemów w praktyce polega na znalezieniu funkcji, która będzie dobrze działać na danych, których nie znamy. Algorytmy PAC-learningowe pozwalają na znalezienie takiej funkcji na podstawie dostępnych danych uczących i minimalizacji ryzyka błędu na danych testowych.

Problemy i ograniczenia

- Co w sytuacji, gdy część wspólna T i F, nie jest zbiorem pustym? Prowadzi to do sprzecznej sytuacji
- Co w sytuacji, gdy część wspólna domknięcie T oraz F, nie jest zbiorem pustym? Ta sytuacja również prowadzi do sprzeczności
- Co w sytuacji jeśli suma domknięcia T oraz F nie wyczerpuje wszystkich możliwych kombinacji wektorów o długośći k?
- Dla pierwszych dwóch sytuacji nie będziemy mogli nic poradzić, lecz w przypadku trzeciej można zastosować pewną strategię

Strategie

- W wcześniej wspomnianym trzecim przypadku możemy obliczyć domknięcie zbioru T w sposób przybliżony
- Mamy do wyboru dwie strategie albo taką która obliczy największe możliwe domknięcie lub drugą, która obliczy najmniejsze
- Należy pamiętać o tym, że wyliczone elementy domknięcia nie mogą pokrywać się z F(przypadek 2 z poprzedniego slajdu)

Strategia cz.2

- Strategia „large” oblicza największe domknięcie T. Wytworzona formuła spełnia, również warunek, że dla każdego f należącego do F jest niespełniona
- Strategia „exact”(najmniejsza liczność domknięcia). Obliczona formuła jest spełniona dla każdego t należącego do T oraz jest niespełniona dla każdego f należącego do zbioru wszystkich wektorów o długości k oprócz domknięcia T(pozostałych)

Minimalna selekcja

- Chcemy uzyskać zbiory domknięcie T oraz F tak, żeby były rozłączne oraz miały jak najmniejszą liczbę koordynatów(rozmiar pojedynczego wektora).
- Problem jest NP-zupełny
- Dlatego też przyjmujemy kilka podejść przybliżonych za pomocą kierunków, zawsze pomijają współrzędne, których usunięcie sprawiłoby, że problem stałby się nierozwiązywalny. Dostępne są następujące kierunki:

Dostępne kierunki

- Od początku – algorytm idzie po koordynatach od początku
- Od końca – algorytm idzie po koordynatach od końca
- Preferuj koordynaty o mniejszej wadze Hamminga, usuwając koordynaty o wysokiej wadze Hamminga(liczba bitów w dwóch ciągach, które różnią się między sobą)
- Preferuj koordynaty o większej wadze Hamminga, usuwając koordynaty o małej wadze Hamminga.
- Usuwaj koordynaty w kolejności losowej
- Opcja bez usuwania koordynatów

Efektywne uczenie formuł

- Uczenie formuł Horna – dla każdego f należącego do F, sprawdza efektywnie czy f nie należy do domknięcia T(bez liczenia domknięcia). Potem oblicza minimalną selekcję domknięcia T oraz F. Domknięcia Horna jest liczone jedną ze wspomnianych wcześniej strategii „large” lub „exact” w zależności od wybranych ustawień

Weryfikacja poprawności

- Sprawdzenie czy f nie należy do domknięcia T przebiega w następujący sposób.
- Ograniczamy rozpatrywane wartości ze zbioru T do wielkości większych leksykograficznie
- Następnie szukamy minimalnego elementu z tego podzbioru w sposób następujący, dokonujemy po kolei koniunkcji kolejnych elementów wektorów, co da nam wektory minimalny
- Jeśli wektor f jest równy tej minimalnej wartości to znaczy, że formuła należy do domknięcia T i jest to przypadek sprzeczny, który należy pominąć

Strategia „large” w szczegółach

- Strategia large tworzy klauzulę po kolei na podstawie wartości ze zbioru F, jeśli f[i](i-ty element wektora z F) ma wartość true to do klauzuli dodajemy zanegowany literał aby na pewno nie była spełniona dla f. Jeśli ma wartość false to ustawiamy wartość neutralną
- Sprawdzamy czy klauzula jest spełniona dla wszystkich T, jeśli tak to dodajemy ja do formuły, jeśli nie to po kolei ustawiamy literały na niezanegowane, tam gdzie i f[i] == false i po każdym ustawieniu sprawdzamy, czy klauzula w takiej konfiguracji jest spełniana przez wszystkie T
- W ten sposób uzyskujemy formułę, która zawsze jest spełniana przez T, a każda klauzula będzie eliminować przynajmniej jeden element z F

Efektywne uczenie formuł

- Uczenie formuł dual Horna – jest wykonywane w bardzo prosty sposób, mianowicie wektory T i F są polaryzowane, czyli wartości wektorów są negowane(0 zmienia się na 1 i na odwrót).
- Następnie dla spolaryzowanych zbiorów obliczana jest formuła horna, która jest polaryzowana(literały nienegowane są zmieniane na negowane), aby uzyskać docelową formułę dualn Horna

Efektywne uczenie formuł

- Uczenie formuł bijunkcyjnych
- Nie istnieje znane rozwiązanie umożliwiające określenie, czy f domknięcia T bijektywnego dla każdego f $\in$ F bez obliczania domknięcia bijektywnego, a więc obliczenie tego jest bardzo czasochłonne i zużywa dużo pamięci

- Algorytm oblicza minimalny przekrój, używając testu przecięcia, a następnie stosując twierdzenie Bakera-Pixleya

Efektywne uczenie formuł

- Uczenie klauzul CNF
- Nauka ogólnej formuły CNF wiąże się z kilkoma wyzwaniami. Zaletą jest to, że otrzymujemy formułę zdaniową w każdym przypadku, pod warunkiem, że $T \cap F = \varnothing$. Wadą jest to, że wygenerowana formuła jest zazwyczaj bardzo duża.
- W przypadku strategii "large" dla każdego fałszywego elementu $f \in F$, MCP system generuje jedyną klauzulę cf, która czyni f fałszywym. Wynikowa formuła φ jest koniunkcją wszystkich klauzul fałszywych cf.
- W przypadku strategii "exact", MCP system używa algorytmu, który w czasie O(|T|k2), gdzie k to arność wektorów w T

Efektywne uczenie formuł

- Uczenie formuł afinicznych nie zostało zaimplementowe.

Postprocessing cz.2

- W przypadku strategii large, naszym głównym celem jest wyprodukowanie formuły φ, która jest falsyfikowana przez każdy wektor $f \in F$
- Jednakże, wynikowa formuła φ może zawierać więcej klauzul niż jest to konieczne
- W celu ograniczenia klauzul stosujemy Set Cover, gdzie klauzula $c \in \varphi$ pokrywa wektor $f \in F$, jeśli f falsyfikuje c
- Jest to znany problem NP-zupełny, lecz istnieje algorytm przybliżający autorstwa Johnsona. Algorytm poszukuje klauzul, które falsyfikują jak największą ilość formuł z F

Optymalizacje

- Do zoptymalizowania działania systemu zastosowano paralelizację, jako że algorytm tworzy n *(n-1) formuł, gdzie n to liczba klas decyzyjnych to obliczenia dla każdej klasy decyzyjnej można wykonywać równolegle, ponieważ są niezależne

Postprocessing

- Formuły mogą zawierać redundantne klauzule oraz literały
- Zostosowano następujące metody:
- Jednostkowa – eliminuje pojedyncze literały
- Jednostkowa i podsumowania klauzul – eliminuje pojedyncze literały oraz całe klauzule.

Opis metod

- Metoda podsumowania klauzul:
- Funkcja wykonuje operację subsumpcji na klauzulach w formule logicznej. Klauzule są porządkowane według długości przed wywołaniem funkcji, a następnie przeszukiwane w kolejności ich występowania w formule. Dla każdej klauzuli sprawdza się, czy istnieje inna klauzula w formule, która jest podzbiorowa wobec niej.
- Metoda jednostkowa:
- W trakcie działania funkcji klauzule zostają podzielone na klauzule jednostkowe oraz pozostałe klauzule, a następnie przeprowadzane jest łączenie ich w procesie wnioskowania jednostkowego, czyli usuwanie literałów, które są już przeciwnie zdefiniowane w innej klauzuli.

Binaryzacja danych

- Jako, że mamy do czynienia z formułami logicznymi to algorytm ten przyjmuje tylko wektory binarne, a więc musiała zostać przeprowadzona konwersja
- Przy konwersji należy uważać, aby wyprodukowane wektory nie były za duże, aby czas wykonania programu nie wzrósł nam za bardzo.
- Proces binaryzacji, nie został opisany w dokumencie, a kod był ciężki w analizie
- Na podstawie przeprowadzonych testów udało się wywnioskować, że np. dla wartości float binaryzacja przebiegała w ten sposób, że jeden literał odpowiadał na pytanie, czy wartość jest w jakimś zakresie

Chat GPT-3 na pomoc

- Oto jak moduł binaryzacji opisał chat gpt-3:
- Dla każdego indeksu "target" funkcja sprawdza typ zmiennej decyzyjnej (np. typ boolowski, typ int, float itp..) i na tej podstawie określa długość ciągu zmiennych decyzyjnych, które będą generowane dla tej zmiennej.
- Następnie funkcja generuje ciąg zmiennych decyzyjnych dla danej zmiennej, który określa przedział wartości, dla którego ta zmienna jest prawdziwa. W zależności od typu zmiennej decyzyjnej, przedział ten może być wyznaczony przez różne kryteria (np. dla typu boolowskiego wartością może być 0 lub 1, dla typu UP wartością może być większa lub równa określonemu punktowi, itp.).

Strategie uczenia

- Do wyboru mamy następujące strategie uczenia
- One vs One czyli naszym zbiorem T jest rozpatrywana klasa decyzyjna, a do F w tym wypadku należy jakaś jedna inna klasa decyzyjna
- One vs All zbiór T bez zmian, natomiast jest on zestawiany z całą resztą formuł z innych klas decyzyjnych
- One vs All bez selekcji – tak samo jak One vs All z tym wyjątkiem, że nie zastosowano metody selekcji
- Selected vs all – wybrana klasa decyzyjna vs reszta

Ewaluacja formuł

- Ewaluacja formuł następuje poprzez obliczenie ilości wartości (TruePositivie, TrueNegative, FalsePositivie, FalseNegative)
- Optymalna sytuacja polegałaby na braku zarówno fałszywie pozytywnych, jak i fałszywie negatywnych wyników. Jeśli jednak wartości te nie są równe zero, może to być spowodowane niewystarczającą licznością danych uczących, błędną binarizacją lub niewystarczającą precyzją danych.

Testy

- System został przetestowany na różnych zbiorach danych z repozytorium uci, wyniki pokazały, że czas wykonania jak i wyniki ewaluacji stoją na bardzo dobrym poziomie

Przykład uruchomienia

- Przykładowe wyniki dla zbioru danych rozpoznających z jakim akcentem mamy do czynienia na podstawie różnych parametrów dźwięku.
-

```
+++ Satisfying: ES
+++ Falsifying: FR GE IT UK US

+++ Formula [116] =
    | (X1<-3||X1>=-2)
  * (X1<-2||X1>=-1)
  * (X1<1||X1>=2)
  * (X1<2||X1>=3)
  * (X2<-2.33333||X2>=-1.27778)
  * (X2<-0.222222||X2>=0.833333)
  * (X1<-5||X1>=-4 + X2<0.833333||X2>=1.88889)
  * (X1<-4||X1>=-3 + X2<-1.27778||X2>=-0.222222)
  * (X1<-4||X1>=-3 + X2<-3.38889||X2>=-2.33333)
  * (X1<-4||X1>=-3 + X2<0.833333||X2>=1.88889)
  * (X1<-1||X1>=0 + X2<-1.27778||X2>=-0.222222)
  * (X1<-1||X1>=0 + X2<0.833333||X2>=1.88889)
  * (X1<-1||X1>=0 + X3<0||X3>=1)
  * (X1<0||X1>=1 + X2<-4.44444||X2>=-3.38889)
  * (X1<0||X1>=1 + X2<-3.38889||X2>=-2.33333)
  * (X1<0||X1>=1 + X2<-1.27778||X2>=-0.222222)
  * (X1<0||X1>=1 + X2<-5.5||X2>=-4.44444)
  * (X1<0||X1>=1 + X2<0.833333||X2>=1.88889)
  * (X1<3||X1>=4 + X2<-1.27778||X2>=-0.222222)
  * (X1<3||X1>=4 + X2<-4.44444||X2>=-3.38889)
  * (X1<3||X1>=4 + X2<-3.38889||X2>=-2.33333)
  * (X1<3||X1>=4 + X2<-5.5||X2>=-4.44444)
```

Metryki formuły

```
3
4   +++ Statistics:
5       ===========
6   +++ true  positive (tp)  = 93
7   +++ true  negative (tn)  = 976
8   +++ false positive (fp)  = 0
9   +++ false negative (fn)  = 0
0   +++ sensitivity    (tpr) = 100     %     [tp / (tp + fn)]
1   +++ miss rate      (fnr) = 0       %     [fn / (fn + tp)]
2   +++ fall-out       (fnr) = 0       %     [fp / (fp + tn)]
3   +++ specificity    (tnr) = 100     %     [tn / (tn + fp)]
4   +++ precision      (ppv) = 100     %     [tp / (tp + fp)]
5   +++ accuracy       (acc) = 100     %     [(tp + tn) / (tp + tn + fp +fn)]
6   +++ F_1 score      (F_1) = 100     %     [tp / (tp + 0.5 * (fp +fn))]
7
```

Koniec

- Dziękuję za uwagę:
- Michał Jakubowski