Wykład 7: Drzewa decyzyjne dla dużych zbiorów danych

Nguyen Hung Son

Funkcja rekurencyjna $buduj\_drzewo(U, dec, \mathbf{T})$:

1: **if** $(kryterium\_stopu(U, dec) = \mathbf{true})$ **then**
2: $\quad \mathbf{T}.etykieta = kategoria(U, dec)$;
3: $\quad$ **return**;
4: **end if**
5: $t := wybierz\_test(U, \mathbf{TEST})$;
6: $\mathbf{T}.test := t$;
7: **for** $v \in R_t$ **do**
8: $\quad U_v := \{x \in U : t(x) = v\}$;
9: $\quad$ utwórz nowe poddrzewo $\mathbf{T}'$;
10: $\quad \mathbf{T}.$gałąź$(v) = \mathbf{T}'$;
11: $\quad buduj\_drzewo(U_v, dec, \mathbf{T}')$
12: **end for**

Funkcje pomocnicze

- Kryterium stopu: Zatrzymamy konstrukcji drzewa, gdy aktualny zbiór obiektów:
  - jest pusty lub
  - zawiera obiekty wyłącznie jednej klasy decyzyjnej lub
  - nie ulega podziale przez żaden test
- Wyznaczenie etykiety zasadą większościową:

$$kategoria(P, dec) = \arg \max_{c \in V_{dec}} |P_{[dec=c]}|$$

  tzn., etykietą dla danego zbioru obiektów jest klasa decyzyjna najliczniej reprezentowana w tym zbiorze.
- Kryterium wyboru testu: heurytyczna funkcja oceniająca testy.

Miary różnorodności zbioru

Każdy zbiór obiektów X ulega podziale na klasy decyzyjne:

$$X = C_1 \cup C_2 \cup ... \cup C_d$$

gdzie $C_i = \{u \in X : dec(u) = i\}$.
Wektor $(p_1, ..., p_r)$, gdzie $p_i = \frac{|C_i|}{|X|}$, nazywamy rozkładem klas decyzyjnych w X.

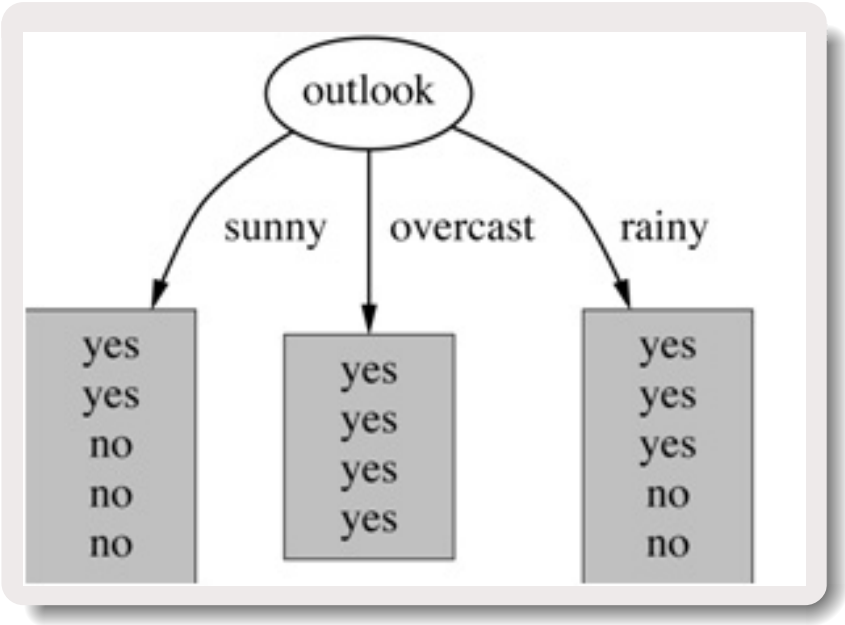
$$Conflict(X) = \sum_{i<j} |C_i| \times |C_j| = \frac{1}{2}\left(|X|^2 - \sum |C_i|^2\right)$$

$$Entropy(X) = -\sum \frac{|C_i|}{|X|} \cdot \log \frac{|C_i|}{|X|} = -\sum p_i \log p_i$$

$$Gini(X) = 1 - \sum p_i^2$$

Ocena funkcji testu

Każdy test t jest oceniony na podstawie informacji zawartych w X, $X_1, ..., X_{n_t}$



Podział zbioru X dokonany przez test t;

Ocena funkcji testu

- Rozróżnialność:

$$disc(t, X) = conflict(X) - \sum conflict(X_i)$$

- Przyrostu informacji (Information gain).

$$Gain(t, X) = Entropy(X) - \sum \frac{|X_i|}{|X|} \cdot Entropy(X_i)$$

- Gini's index

$$G(t, X) = Gini(X) - \sum \frac{|X_i|}{|X|} \cdot Gini(X_i)$$

- Kara za zbyt drobny podział, np. gain ratio

$$Gain\_ratio = \frac{Gain(t, X)}{-\sum_{i=1}^{r} \frac{|X_i|}{|X|} \cdot \log \frac{|X_i|}{|X|}}$$

Przykład

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

Słabości standardowego algorytmu:

- Każdy węzeł jest skojarzony z podzbiorem danych: ograniczenie pamięciowe
- Wyznaczenie najlepszego podziału wymaga wielokrotnego sortowania danych: czasochłonne
  - Dany atrybut rzeczywisty a i zbiór możliwych cięć $(c_1, c_2, ...c_N)$, najlepszy test (a, c_i) można znaleźć w czasie $\Omega(N)$
  - Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego testu jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych
- Wniosek: szukanie najlepszego podziału jest kosztowne, jeśli atrybut zawiera dużo różnych wartości.

Charaterystyka algorytmu SPRINT

- Nadaje się dla danych częściowo umieszczonych na dysku
- Używa się techniki pre-sortowania w celu przyspieszenia procesu obliczenia na atrybutach rzeczywistych;
- Dane są sortowane tylko raz przed obliczeniem
- Łatwo można zrównoleglić

Struktura danych w SPRINT

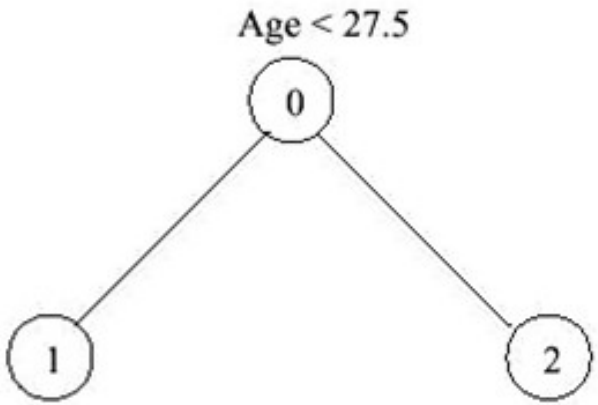
- Każdy atrybut ma swoją listę wartości
- Każdy element listy ma trzy pole:
  - *wartość atrybutu,*
  - *numer klasy* i
  - *rid* (numer obiektu w zbiorze danych)
- Rzeczywiste atrybuty są uporządkowane (tylko raz przy utworzeniu)
- Na początku listy są stowarzyszone z korzeniem drzewa
- Kiedy węzeł podlega podziale, listy są podzielone i są skojarzone z odpowiednimi następnikami
- Listy są zapisane na dysku w razie potrzeby

Przykład

Attribute lists for node 0

Age	Class	Tid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Car Type	Class	Tid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5



Attribute lists for node 1

Age	Class	Tid
17	High	1
20	High	5
23	High	0

Car Type	Class	Tid
family	High	0

Attribute lists for node 2

Age	Class	Tid
32	Low	4
43	High	2
68	Low	3

Car Type	Class	Tid
sports	High	2

Struktura danych w SPRINT

- SPRINT używa:
  - indeksu Gini do oceny jakości podziału
  - testu typu $(a \leq c)$ dla atrybutów rzeczywistych
  - testu typu $(a \in V)$ dla atrybutów symbolicznych
- Histogram: rozkład klas decyzyjnych zbadanego zbioru danych
- Dla atrybutu rzeczywistego dwa histogramy:
  C_{below} : histogram dla danych poniżej wartości progowej
  C_{above} : histogram dla danych powyżej wartości progowej
- Dla atrybutu symbolicznego jeden histogram zwany count matrix

Przykład

Car Type	*Class*	*rid*
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	high	5

Punkt podziału

Age	*Class*	*rid*
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Count matrix

	H	L
family	2	1
sports	2	0

Histogram klas

	H	L
C_{below}	3	0
C_{above}	1	2

Outline

Wyznaczanie podziału atrybutu rzeczywistego

Attribute List

Age	Class	tid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Position of cursor in scan

← position 0

← position 3

← position 6

State of Class Histograms

cursor position 0:

	H	L
C_{below}	0	0
C_{above}	4	2

cursor position 3:

	H	L
C_{below}	3	0
C_{above}	1	2

cursor position 6:

	H	L
C_{below}	4	2
C_{above}	0	0

Wyznaczanie podziału atrybutu symbolicznego

Lista wartości ***CarType***

Car Type	Class	rid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	high	5

Count Matrix

	H	L
family	2	1
sports	2	0
truck	0	1

1. Wyznacz macierz rozkładu klas obiektów w danym węźle
2. Używając algorytmu aproksymacyjnego (w SLIQ) *wyznacz* podzbiór wartości $V \subseteq D_a$ t. żeby test ($a \in V$) był optymalny

Outline

Główna idea

- Każda lista jest podzielona na dwie listy
- **Atrybut zawierający test:** Podziel wartości listy zgodnie z testem
- **Atrybut niewierający test:**
  - Nie można korzystać z informacji w funkcji testu.
  - Skorzystaj z *rid*
  - Skorzystaj z tablicy haszującej
- **Przy podziale atrybutu zawierający test:** wstaw rid rekordów do tablicy haszującej.
- **Tablica haszująca:** informacje o tym do którego poddrzewa rekord został przeniesiony.
- **Algorytm:**
  - Przeglądaj kolejny rekord listy
  - Dla każdego rekordu wyznacz (na podstawie tablicy haszującej) poddrzewo, do którego rekord ma być przeniesiony

Problem: zbyt duża tablica haszująca

Algorytm:

- Krok 1: Podziel zbiór wartości atrybutu testującego na małe porcje tak, żeby tablica haszująca mieściła się w pamięci
- Krok 2: Dla każdej porcji
  - Podziel rekordy atrybutu testującego do właściwego podrzewa
  - Buduj tablicę haszującej
  - Przeglądaj kolejny rekord atrybutu nietestującego i przynieś go do odpowiedniego poddrzewa jeśli rekord występuje w tablicy haszującej
- Krok 3: Jeśli wszystkie rekordy zostały przydzielone do poddrzew **stop**, wpp. **idź do** krok 2

Outline

Równoległy SPRINT

- Listy wartości atrybutów są równo podzielone
- **Atrybut rzeczywisty:** sortuj zbiór wartości i podziel go na równe przedziały
- **Atrybut numeryczny:** podziel według rid
- Każdy procesor ma jedną część każdej listy

Szukanie najlepszego podziału

- Dla atrybutu rzeczywistego:
  - Każdy procesor ma przedział wartości atrybutu
  - Każdy procesor inicjalizuje C_{below} i C_{above} uwzględniając rozkład klas w innych procesorach
  - Każdy procesor przegląda swoją listę i wyznacza najlepszą lokalną wartość progową
  - Procesory komunikują się w celu znalezienia globalnie najlepszego cięcia
- Dla atrybutu symbolicznego:
  - Każdy procesor buduje lokalne count matrix i wysyła wynik do centralnego procesora
  - Centralny procesor oblicza globalny count matrix
  - Procesory wyznaczają najlepszy podział na podstawie globalnego count matrix

Przykład

Procesor 0

Age	*Class*	*rid*
17	High	1
20	High	5
23	High	0

Car Type	*Class*	*rid*
family	High	0
sports	High	1
sports	High	2

Procesor 1

Age	*Class*	*rid*
32	Low	4
43	High	2
68	Low	3

Car Type	*Class*	*rid*
family	Low	3
truck	Low	4
family	high	5

Dokonanie podziału

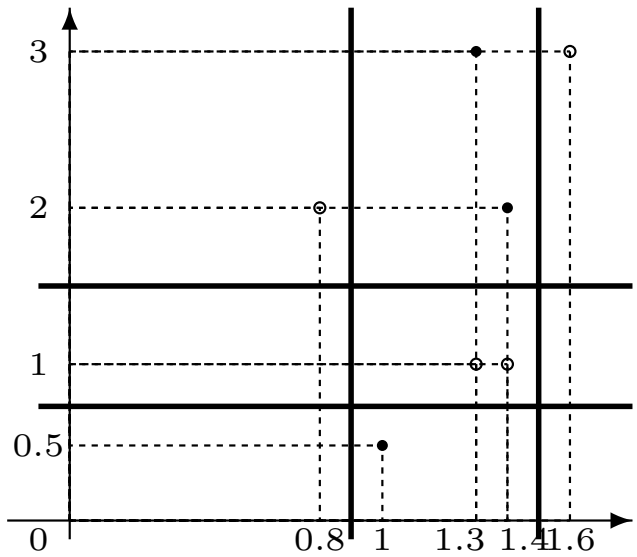
- Podział atrybutu zawierający test: Każdy procesor wyznacza poddrzewa, do których rekordy w lokalnej liście będą przeniesione
- Procesory wymieniają ze sobą informacje $\langle rids, poddrzewo \rangle$
- Podział pozostałych atrybutów: Po otrzymaniu informacji ze wszystkich procesorów każdy procesor buduje tablicę haszującą i wykonuje podziały dla pozostałych atrybutów

Wady algorytmu SPRINT

- Dodatkowe struktury danych
- Nieefektywny jeśli atrybut ma dużo wartości
- Nie wykorzystuje mocnych narzędzi systemów baz danych

Dyskretyzacja

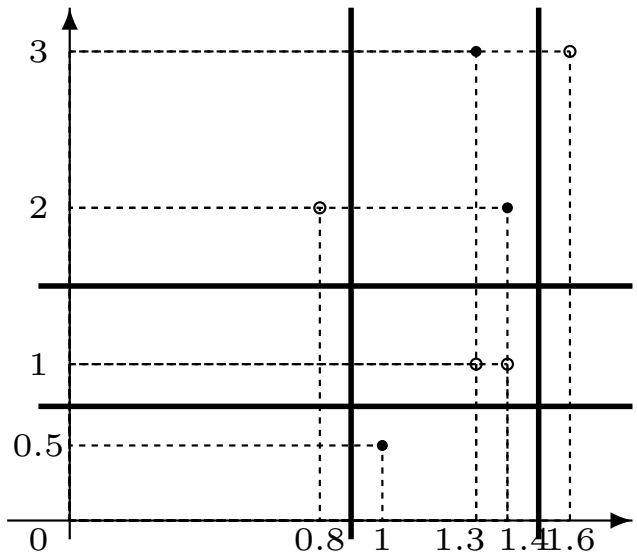
$\mathbb{S}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1



- Zmienne Boolowskie: $p_1^a, p_2^a, p_3^a, p_4^a, \quad p_1^b, p_2^b, p_3^b$
  odpowiadają$(a, 0.9), (a, 1.15), (a, 1.35), (a, 1.5), (b, 0.75), (b, 1.5), (b, 2.5)$;

Dyskretyzacja

$\mathbb{S}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1

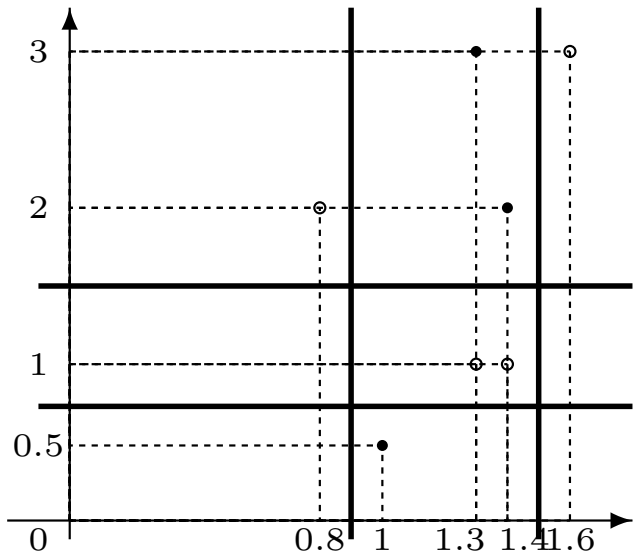


- Zmienne Boolowskie: $p_1^a, p_2^a, p_3^a, p_4^a, \quad p_1^b, p_2^b, p_3^b$
  odpowiadają$(a, 0.9), (a, 1.15), (a, 1.35), (a, 1.5), (b, 0.75), (b, 1.5), (b, 2.5)$;
- Funkcja kodująca problem dyskretyzacji

$$\Phi_{\mathbb{S}} = \left(p_1^a + p_1^b + p_2^b\right)\left(p_1^a + p_2^a + p_3^b\right)\left(p_1^a + p_2^a + p_3^a\right)\left(p_2^a + p_3^a + p_1^b\right)$$
$$p_2^b\left(p_2^a + p_2^b + p_3^b\right)\left(p_2^a + p_3^a + p_4^a + p_1^b + p_2^b + p_3^b\right)\left(p_3^a + p_4^a\right)$$
$$\left(p_4^a + p_3^b\right)\left(p_2^a + p_1^b\right)\left(p_2^b + p_3^b\right)\left(p_3^a + p_2^b\right)$$

Dyskretyzacja

$\mathbb{S}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1



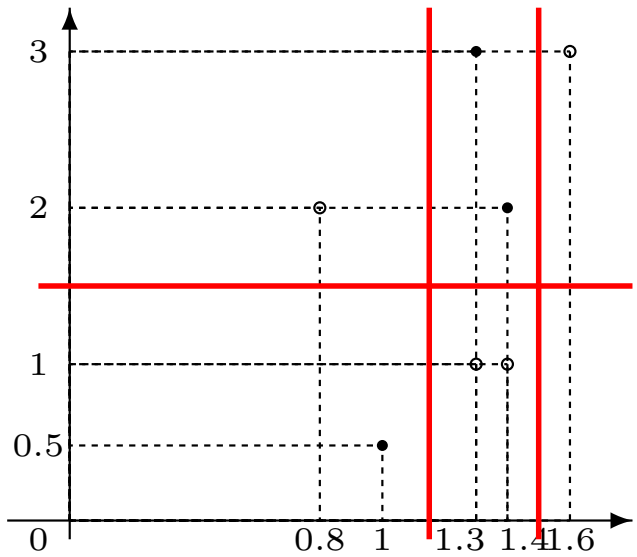
- Zmienne Boolowskie: $p_1^a, p_2^a, p_3^a, p_4^a, \quad p_1^b, p_2^b, p_3^b$
  odpowiadają$(a, 0.9), (a, 1.15), (a, 1.35), (a, 1.5), (b, 0.75), (b, 1.5), (b, 2.5)$;
- Funkcja kodująca problem dyskretyzacji

$$\Phi_{\mathbb{S}} = \left(p_1^a + p_1^b + p_2^b\right)\left(p_1^a + p_2^a + p_3^b\right)\left(p_1^a + p_2^a + p_3^a\right)\left(p_2^a + p_3^a + p_1^b\right)$$
$$p_2^b\left(p_2^a + p_2^b + p_3^b\right)\left(p_2^a + p_3^a + p_4^a + p_1^b + p_2^b + p_3^b\right)\left(p_3^a + p_4^a\right)$$
$$\left(p_4^a + p_3^b\right)\left(p_2^a + p_1^b\right)\left(p_2^b + p_3^b\right)\left(p_3^a + p_2^b\right)$$

- Po redukcji: $\Phi_{\mathbb{S}} = p_2^a p_4^a p_2^b + p_2^a p_3^a p_2^b p_3^b + p_3^a p_1^b p_2^b p_3^b + p_1^a p_4^a p_1^b p_2^b$.

Dyskretyzacja

$\mathbb{S}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1



- Zmienne Boolowskie: $p_1^a, p_2^a, p_3^a, p_4^a, \quad p_1^b, p_2^b, p_3^b$ odpowiadają $(a, 0.9), (a, 1.15), (a, 1.35), (a, 1.5), (b, 0.75), (b, 1.5), (b, 2.5)$;
- Funkcja kodująca problem dyskretyzacji

$$\Phi_{\mathbb{S}} = \left(p_1^a + p_1^b + p_2^b\right)\left(p_1^a + p_2^a + p_3^b\right)\left(p_1^a + p_2^a + p_3^a\right)\left(p_2^a + p_3^a + p_1^b\right)$$
$$p_2^b\left(p_2^a + p_2^b + p_3^b\right)\left(p_2^a + p_3^a + p_4^a + p_1^b + p_2^b + p_3^b\right)\left(p_3^a + p_4^a\right)$$
$$\left(p_4^a + p_3^b\right)\left(p_2^a + p_1^b\right)\left(p_2^b + p_3^b\right)\left(p_3^a + p_2^b\right)$$

- Po redukcji: $\Phi_{\mathbb{S}} = p_2^a p_4^a p_2^b + p_2^a p_3^a p_2^b p_3^b + p_3^a p_1^b p_2^b p_3^b + p_1^a p_4^a p_1^b p_2^b$.

MD-heuristics

- Funkcja Boolowska kodująca problem dyskretyzacji posiada $O(nk)$ zmiennych i $O(n^2)$ klauzuli, gdzie n jest liczbą obiektów, k jest liczbą atrybutów.

MD-heuristics

- Funkcja Boolowska kodująca problem dyskretyzacji posiada $O(nk)$ zmiennych i $O(n^2)$ klauzuli, gdzie n jest liczbą obiektów, k jest liczbą atrybutów.
- W algorytmie zachłannej, preferujemy cięcia, które rozróżniają najwięcej par obiektów.

MD-heuristics

- Funkcja Boolowska kodująca problem dyskretyzacji posiada $O(nk)$ zmiennych i $O(n^2)$ klauzuli, gdzie n jest liczbą obiektów, k jest liczbą atrybutów.
- W algorytmie zachłannej, preferujemy cięcia, które rozróżniają najwięcej par obiektów.
- Taki algorytm nazywamy "heurystyką MD". Opracowane są wersje globalne i lokalne.

MD-heuristics

- Funkcja Boolowska kodująca problem dyskretyzacji posiada $O(nk)$ zmiennych i $O(n^2)$ klauzuli, gdzie n jest liczbą obiektów, k jest liczbą atrybutów.
- W algorytmie zachłannej, preferujemy cięcia, które rozróżniają najwięcej par obiektów.
- Taki algorytm nazywamy "heurystyką MD". Opracowane są wersje globalne i lokalne.
- Bezpośrednia implementacja heurystyki MD (z użyciem funkcji Boolowskiej) wymaga $O(n^3k)$ obliczeń w każdym kroku.

MD-heuristics

- Funkcja Boolowska kodująca problem dyskretyzacji posiada $O(nk)$ zmiennych i $O(n^2)$ klauzuli, gdzie n jest liczbą obiektów, k jest liczbą atrybutów.
- W algorytmie zachłannej, preferujemy cięcia, które rozróżniają najwięcej par obiektów.
- Taki algorytm nazywamy "heurystyką MD". Opracowane są wersje globalne i lokalne.
- Bezpośrednia implementacja heurystyki MD (z użyciem funkcji Boolowskiej) wymaga $O(n^3k)$ obliczeń w każdym kroku.
- Można realizować heurystykę MD w czasie $O(nk \log n|\mathbf{P}|)$, gdzie $\mathbf{P}$ jest zbiorem cięć znalezionych przez algorytm

Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;

Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;
- Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego cięcia jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych

Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;
- Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego cięcia jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych
- Przedstawione w rozprawie 3 techniki pozwalające wyznaczyć najlepsze cięcie za pomocą $O(d \log N)$ zapytań:

Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;
- Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego cięcia jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych
- Przedstawione w rozprawie 3 techniki pozwalające wyznaczyć najlepsze cięcie za pomocą $O(d \log N)$ zapytań:
  - Eliminacja cięć nie będących brzegami;

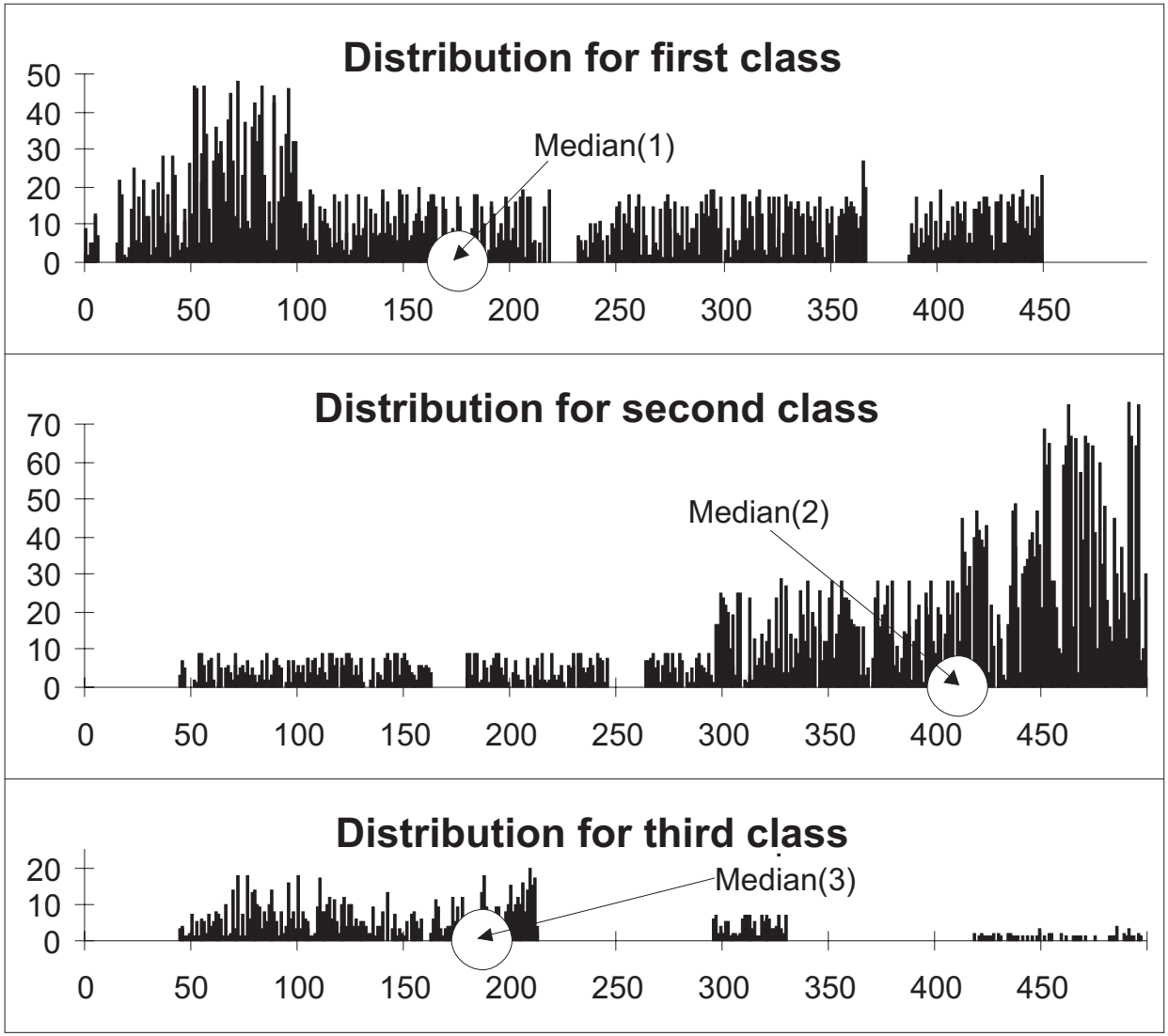
Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;
- Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego cięcia jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych
- Przedstawione w rozprawie 3 techniki pozwalające wyznaczyć najlepsze cięcie za pomocą $O(d \log N)$ zapytań:
  - Eliminacja cięć nie będących brzegami;
  - Obcinanie ogonków;

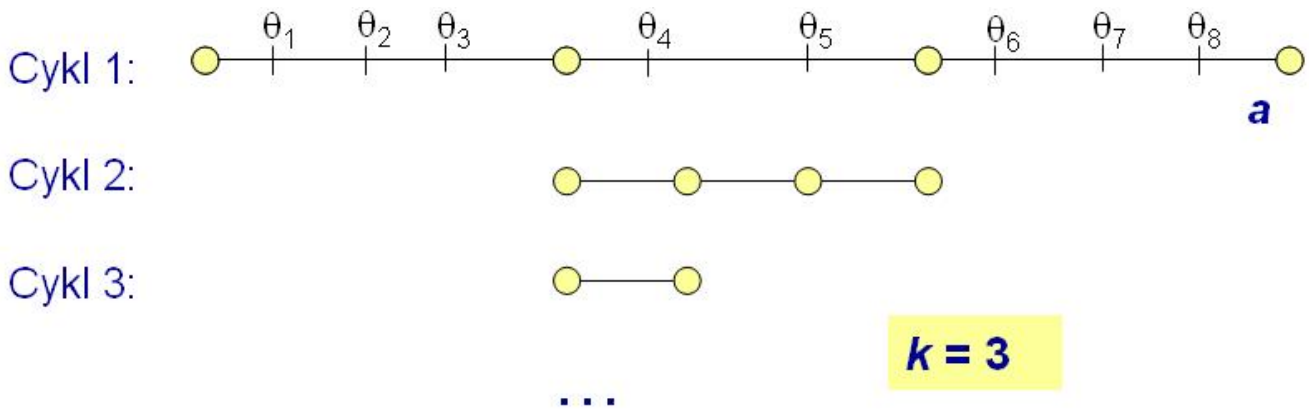
Dyskretyzacja za pomocą zapytań SQL

- Dany atrybut rzeczywisty a i zbiór możliwych wartości progowych $(t_1, t_2, ...t_N)$, najlepszy cięcie (a, t_{Best}) można znaleźć w czasie $O(N)$;
- Minimalna liczba prostych zapytań SQL potrzebna do szukania najlepszego cięcia jest $O(dN)$, gdzie d jest liczbą klas decyzyjnych
- Przedstawione w rozprawie 3 techniki pozwalające wyznaczyć najlepsze cięcie za pomocą $O(d \log N)$ zapytań:
  - Eliminacja cięć nie będących brzegami;
  - Obcinanie ogonków;
  - Strategia “dziel i rządź”

Obcinanie ogonków



Dziel i rządź



- Podziel zbiór wartości atrybutu na k przedziałów
- Oceń przedziały, aby zgadnąć który z przedziałów zawiera najlepsze cięcie

$$Eval([c_L, c_R]) = \frac{W(c_L) + W(c_R) + conflict([c_L, c_R])}{2} + \Delta$$

- Wybierz najlepszy przedział (lub odrzuć słabe przedziały);
- Powtórz proces dla wybranego przedziału (wybranych przedziałów);

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;
  - Przedstawione są metody zarówno na pojedynczych atrybutach jak i na wszystkich atrybutach.

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;
  - Przedstawione są metody zarówno na pojedynczych atrybutach jak i na wszystkich atrybutach.
  - Eksperymenty pokazują, że znalezione cięcia są dość bliskie optymalnych;

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;
  - Przedstawione są metody zarówno na pojedynczych atrybutach jak i na wszystkich atrybutach.
  - Eksperymenty pokazują, że znalezione cięcia są dość bliskie optymalnych;
- Opracowane są podobne metody dla miary entropii;

ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;
  - Przedstawione są metody zarówno na pojedynczych atrybutach jak i na wszystkich atrybutach.
  - Eksperymenty pokazują, że znalezione cięcia są dość bliskie optymalnych;
- Opracowane są podobne metody dla miary entropii;
- Można tę metodę wykorzystać do konstrukcji drzew decyzyjnych z dużych zbiorów danych;

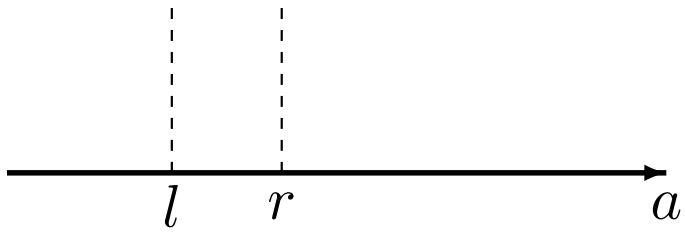
ABR w problemie dyskretyzacji

- "Dziel i rządź" jest kolejnym przykładem podejścia ABR:
  - Analizujemy przedziały, czyli zbiory zmiennych Boolowskich (lub zbiory odpowiednich cięć) zamiast pojedynczych cięć;
  - Wykorzystane są cechy związane z porządkiem liniowym na cięciach;
  - Przedstawione są metody zarówno na pojedynczych atrybutach jak i na wszystkich atrybutach.
  - Eksperymenty pokazują, że znalezione cięcia są dość bliskie optymalnych;
- Opracowane są podobne metody dla miary entropii;
- Można tę metodę wykorzystać do konstrukcji drzew decyzyjnych z dużych zbiorów danych;
- ... i do znalezienia "elastycznych cięć".

Soft cuts and soft DT

A soft cut is any triple $p = \langle a, l, r \rangle$, where

- $a \in A$ is an attribute,
- $l, r \in \Re$ are called the left and right bounds of p ;
- the value $\varepsilon = \frac{r-l}{2}$ is called the uncertain radius of p.
- We say that a soft cut p discerns a pair of objects x_1, x_2 if $a(x_1) < l$ and $a(x_2) > r$.



- The intuitive meaning of $p = \langle a, l, r \rangle$:
  - *there is a real cut somewhere between* l *and* r.
  - *for any value* $v \in [l, r]$ *we are not able to check if* v *is either on the left side or on the right side of the real cut.*
  - $[l, r]$ *is an uncertain interval of the soft cut* p.
  - *normal cut can be treated as soft cut of radius* 0.

Soft Decision Tree

- The test functions can be defined by soft cuts
- Here we propose two strategies using described above soft cuts:
  - *fuzzy decision tree*: any new object u can be classified as follows:
    - For every internal node, compute the probability that u turns left and u turns right;
    - For every leave L compute the probability that u is reaching L;
    - The decision for u is equal to decision labeling the leaf with largest probability.
  - *rough decision tree*: in case of uncertainty
    - Use both left and right subtrees to classify the new object;
    - Put together their answer and return the answer vector;
    - Vote for the best decision class.

STANDARD ALGORITHM FOR BEST CUT

- For a given attribute a and a set of candidate cuts $\{c_1, ..., c_N\}$, the best cut (a, c_i) with respect to given heuristic measure

$$F : \{c_1, ..., c_N\} \to \mathbb{R}^+$$

  can be founded in time $\Omega(N)$.

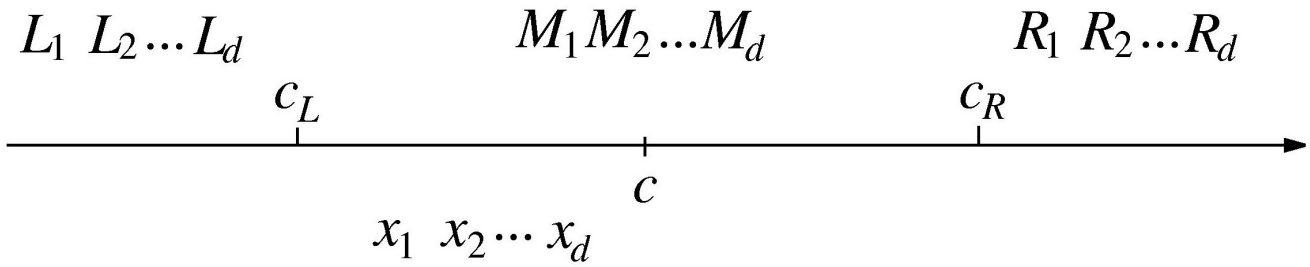
- The minimal number of simple SQL queries of form
  `SELECT COUNT`
  `FROM datatable`
  `WHERE (a BETWEEN` c_L `AND` c_R`) GROUPED BY d.`
  necessary to find out the best cut is $\Omega(dN)$

OUR PROPOSITIONS FOR SOFT CUTS

- Tail cuts can be eliminated
- Divide and Conquer Technique

Divide and Conquer Technique

- The algorithm outline:
  1. *Divide the set of possible cuts into k intervals*
  2. *Chose the interval to which the best cut may belong with the highest probability.*
  3. *If the considered interval is not STABLE enough then Go to Step 1*
  4. *Return the current interval as a result.*
- The number of SQL queries is $O(d \cdot k \log_k n)$ and is minimum for $k = 3$;
- How to define the measure evaluating the quality of the interval $[c_L; c_R]$?



- This measure should estimate the quality of the best cut from $[c_L; c_R]$

Discernibility measure:

We construct estimation measures for intervals in four cases:

	Discernibility measure	Entropy Measure
Independency assumption	?	?
Dependency assumption	?	?

Under **dependency assumption**, i.e.

$$\frac{x_1}{M_1} \simeq \frac{x_2}{M_2} \simeq ... \simeq \frac{x_d}{M_d} \simeq \frac{x_1 + ... + x_d}{M_1 + ... + M_d} = \frac{x}{M} =: t \in [0, 1]$$

discernibility measure for $[c_L; c_R]$ can be estimated by:

$$\frac{W(c_L) + W(c_R) + conflict(c_L; c_R)}{2} + \frac{[W(c_R) - W(c_L)]^2}{conflict(c_L; x_R)}$$

Under **dependency assumption**, i.e. $x_1, ..., x_d$ are independent random variables with uniform distribution over sets $\{0, ..., M_1\}$, ..., $\{0, ..., M_d\}$, respectively.

- The mean $E(W(c))$ for any cut $c \in [c_L; c_R]$ satisfies

$$E(W(c)) = \frac{W(c_L) + W(c_R) + conflict(c_L; c_R)}{2}$$

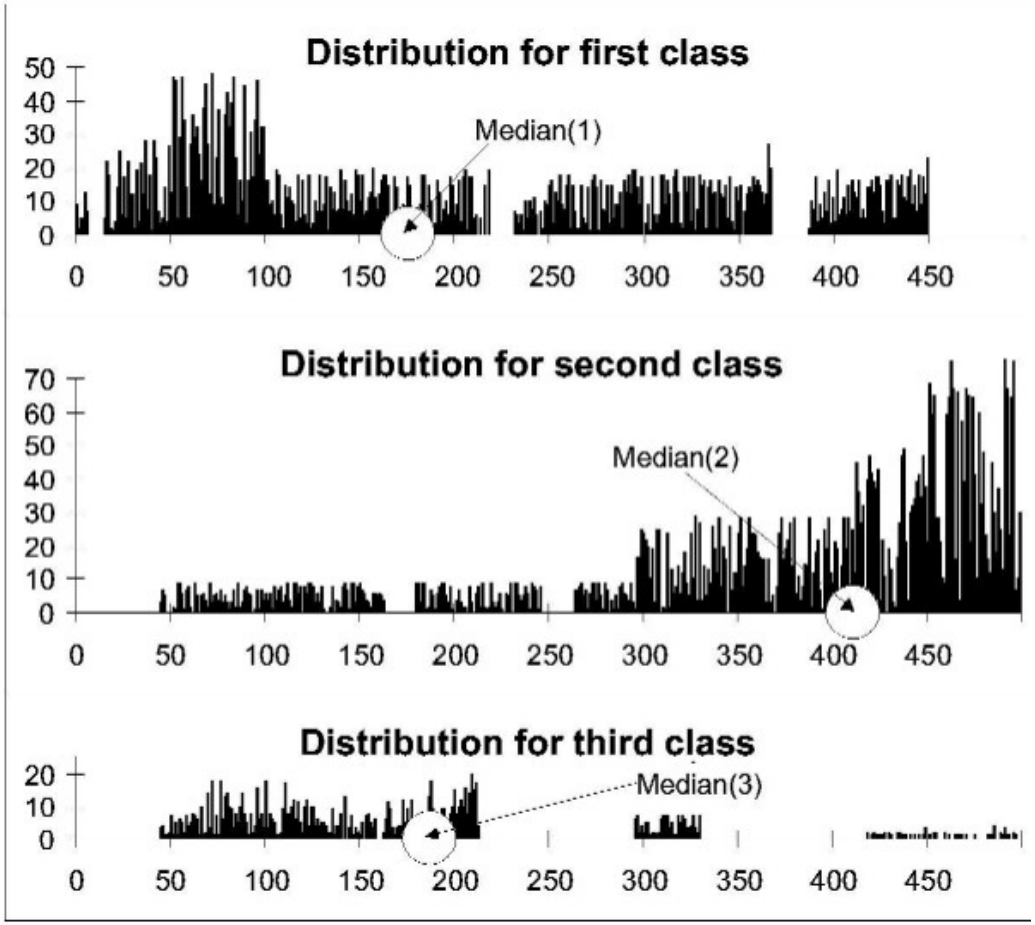
- and for the standard deviation of $W(c)$ we have

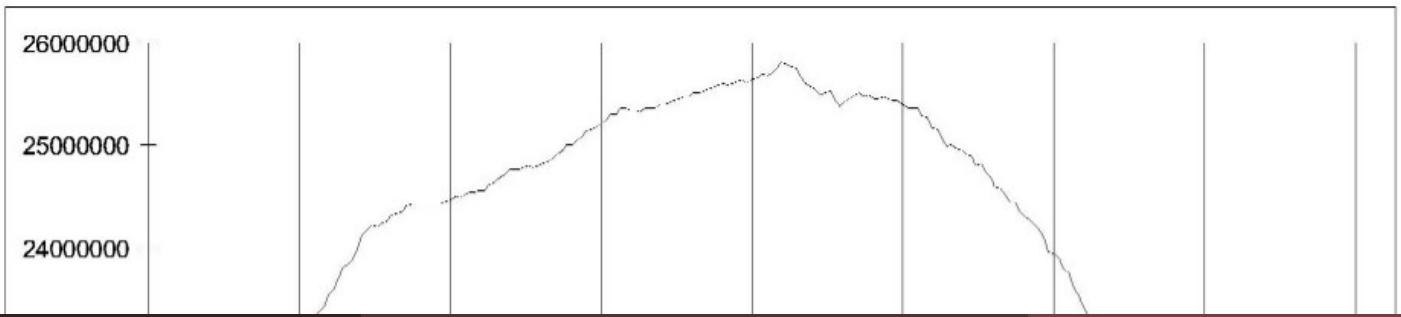
$$D^2(W(c)) = \sum_{i=1}^{n} \left[\frac{M_i(M_i + 2)}{12} \left(\sum_{j \neq i} (R_j - L_j) \right)^2 \right]$$

- One can construct the measure estimating quality of the best cut in $[c_L; c_R]$ by

$$\boxed{Eval\left([c_L; c_R], \alpha\right) = E(W(c)) + \alpha\sqrt{D^2(W(c))}}$$

Example


Distribution for first class
Median(1)
Distribution for second class
Median(2)
Distribution for third class
Median(3)


26000000
25000000
24000000

Conclusions

- Soft cuts as a novel discretization concept;
- Soft decision tree;
- Efficient method for construction of soft cuts from large data (one can reduce the number of simple queries from $O(N)$ to $O(\log N)$ to construct the partition very close to the optimal one).