Aprendizado Descritivo

Aula 12 – Algoritmos evolucionários para descoberta de subgrupos

Professor Renato Vimieiro

DCC/ICEx/UFMG

Introdução

- Como vimos, as abordagens heurísticas são necessárias em situações em que os algoritmos exaustivos falham
- Frequentemente, essa situação ocorre em bases de dados com muitos atributos (alta dimensionalidade)
- Nesses casos, o espaço de busca cresce muito rapidamente, mesmo considerando linguagens com seletores mais simples
- Um dos problemas do beam search é que o resultado final é bastante redundante, apesar da medidas implementadas para combater a redundância

Introdução

- A figura abaixo mostra a cobertura dos 100 melhores subgrupos encontrados em uma base de dados de benchmark
- Como pode ser visto, existem essencialmente duas classes de padrões
  - Uma do 1-60 que cobrem uma parte das amostras
  - Outra do 60-100 que cobrem uma parte disjunta
  - Mas os subgrupos de cada parte cobrem essencialmente os mesmos objetos

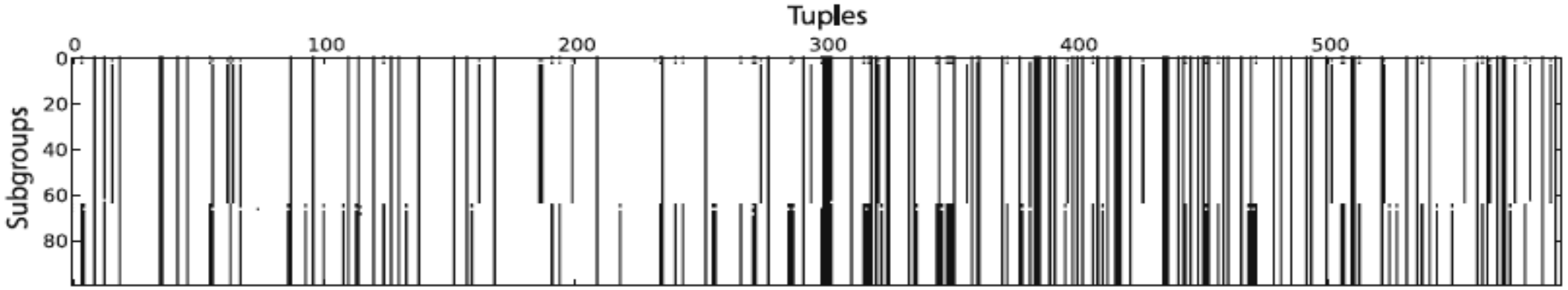

Fig. extraída de Van Leeuwen e Knobbe (2012)

Introdução

- Usando o exemplo de van Leeuwen e Knobbe (2012), assuma uma base de dados com os atributos à esquerda, sendo os melhores individuais como na coluna do meio, e segundo nível do beam search com beam width = 4 à direita
- Percebemos que, já no segundo nível, o beam é dominado por combinações dos melhores seletores

fatalities	positive integer
nature	*{fatal, injured, damage only}*
time	*{day, night}*
cost	positive real

1.	$fatalities \geq 1$
2.	$nature = fatal$
3.	$fatalities \geq 2$
4.	$nature \neq damage\ only$
5.	$time = night$
...	...

1.	$fatalities \geq 1 \wedge nature = fatal$
2.	$fatalities \geq 1 \wedge nature \neq damage\ only$
3.	$fatalities \geq 1 \wedge fatalities \geq 2$
4.	$nature = fatal \wedge cost \geq 123.4$
...	...

Introdução

- Assim, o grande esforço computacional dispendido na busca dos padrões é usado para encontrar combinações dos melhores seletores
- Outra abordagem muito usada em problemas de otimização são heurísticas baseadas em computação natural (em particular, abordagens evolucionárias)
- Abordagens evolucionárias usam metáforas da natureza para construção de algoritmos heurísticos
  - Nesse caso, a teoria de evolução das espécies
- Esses algoritmos possuem aplicações em vários contextos, obtendo bons resultados por sua capacidade de balancear exploração e explotação do espaço de busca
- Vamos ver dois métodos para descoberta de subgrupos baseados em algoritmos genéticos

Algoritmos Genéticos

- Os algoritmos genéticos (GA) se baseiam nos processos biológicos de seleção natural para evolução de espécies
- Intuitivamente, os indivíduos de uma população se cruzam, trocando material genético, o qual pode sofrer mutações para produzir novos indivíduos na população
- Considerando-se que os recursos disponíveis são limitados, apenas os indivíduos mais aptos sobrevivem, passando seu material genético para as novas gerações
- A heurística computacional se baseia nesses princípios para navegar no espaço de busca

Algoritmos Genéticos

- O primeiro passo na construção de um algoritmo genético é definir a representação dos cromossomos dos indivíduos
  - O mais comum é a representação binária, em que cada bit (gene) representa a inclusão ou exclusão de um valor
  - Outras representações mais complexas também podem ser usadas
- O segundo passo é definir uma função de aptidão (fitness), que corresponde à função objetivo que queremos otimizar
- Em seguida, o algoritmo entra em um ciclo onde a população evolui até que um critério de parada seja alcançado

Algoritmos genéticos

- De uma forma geral, os GAs possuem a seguinte estrutura:

```
Início
   t <- 1
   Inicializar a população P(t)
   Avaliar a população P(t)
   enquanto não atingir critério de parada
        Selecionar indivíduos para reprodução em P(t)
        Aplicar operadores genéticos para produzir população P(t+1)
        Avaliar P(t+1)
        t <- t+1
   fim enquanto
Fim
```

Algoritmos Genéticos

- Existem diferentes abordagens para a seleção dos indivíduos que serão usados na reprodução
  - Seleção proporcional ao fitness (método da roleta)
  - Seleção por torneio
- Os demais operadores genéticos são cruzamento e mutação
  - Esses operadores são aplicados com probabilidades estabelecidas pelo usuário
- No cruzamento, um par de indivíduos selecionados pelo operador de seleção trocam seu material genético para construir dois novos indivíduos
  - Existem diferentes formas de combinar esses indivíduos, algumas sendo específicas para alguns tipos de representação
- A mutação altera de forma aleatória um ou mais genes do indivíduo, gerando uma nova solução (indivíduo)
  - Esse operador também é desenhado conforme a aplicação em questão
- Após a geração dos novos indivíduos, a próxima geração é selecionada para continuar a evolução

SSDP

- O algoritmo Simple Search Discriminative Patterns (SSDP) foi proposto por Pontes, Vimieiro e Ludermir em 2016
- O SSDP é uma heurística evolucionária para encontrar top-k subgrupos em bases de dados de alta dimensionalidade
- O algoritmo foi desenhado especificamente para encontrar padrões em bases com essa característica, onde, os outros métodos do estado da arte falhavam
- Dado o grande número de atributos, os autores optaram por simplificar a linguagem de descrição, considerando apenas seletores da forma atributo=valor
  - São aceitos somente dados categóricos

SSDP

- Quanto ao atributo alvo, inicialmente foi previsto somente dados categóricos. Contudo, não há restrições caso a função de fitness seja ajustada para medidas de qualidade numéricas
- Como a ideia do algoritmo era ser simples de usar, vários parâmetros comuns a algoritmos genéticos foram ajustados automaticamente
  - A taxa de mutação e cruzamento são adaptativas e determinadas pelo algoritmo
  - O tamanho da população é fixo, determinado pelo número de seletores da base
  - O usuário tem que determinar o valor de k, e a função de qualidade desejada

SSDP

- O algoritmo mantém dois conjuntos de subgrupos paralelos ao longo da execução
  - Os k melhores subgrupos
  - A população atual do GA
- A população é inicializada com todos os possíveis seletores
  - Isso permite que mais soluções candidatas sejam exploradas pelo algoritmo, uma vez que o espaço de busca é muito grande
- Como os primeiros indivíduos são seletores únicos, houve a necessidade de se criar um operador de cruzamento específico para a primeira geração
  - Esse operador é equivalente ao gerador de candidatos do Apriori
- A partir da segunda geração, foi utilizado cruzamento uniforme
- O operador de mutação consiste em 3 ações distintas equiprováveis
  - Troca de um seletor por outro
  - Remoção de um seletor
  - Inclusão de um seletor
- O operador de seleção é torneio binário

SSDP

- A cada iteração, os k melhores indivíduos **relevantes** tentam substituir os atuais subgrupos em Pk
- Caso não haja melhoria, a taxa de cruzamento é decrementada e mutação aumenta em 0.2
  - As taxas de mutação e cruzamento devem sempre somar a 1
- Se a mutação chegar em 1 e não houver melhora por 3 gerações, a população é reinicializada
  - 10% aleatória com os seletores dos k melhores subgrupos
  - 90% aleatória contendo de 1 à média de seletores por indivíduo na população atual
- Se a população for reinicializada por 3 vezes, a busca é interrompida

SSDP

Require: k, *metricEvaluation*
  $P \longleftarrow \{\{i_1\}, \{i_2\}, \ldots, \{i_{|I|}\}\}$
  $P_k \longleftarrow kBestRelevants(P)$
  reinializationCount $\longleftarrow 0$
  mutationRate $\longleftarrow 0.4$
  crossoverRate $\longleftarrow 0.6$
  while *reinializationCount* < 2 bf do
    while P_k not improve three consecutive generations keeping
  mutationRate == 1.0 **do**
      if generation == 1 **then**
        $P_{new} \longleftarrow crossoverAND(P)$
      else {generation > 1}
        $P_{new} \longleftarrow evolutionaryOperator(P, mutationRate, crossoverRate)$
      end if
      $P^* \longleftarrow best(P, P_{new})$
      $P_k \longleftarrow kBestRelevants(P_k, P_*)$
      update(*mutationRate*, *crossoverRate*)
      $P \longleftarrow P^*$
    end while
    reinializationCount ++
    $P \longleftarrow$ restart
  end while
  return P_k

SSDP

Name	$\|D\|$	$\|D^+\|$	$\|D^-\|$	Attributes
Alon	62	40	22	2000
Borovecki	31	17	14	22,283
Burczynski	127	59	68	22,283
Chiaretti	128	74	54	12,625
Chin	118	75	43	22,215
Chowdary	104	62	42	22,283
Christensen	217	113	104	1413
Golub	72	47	25	7129
Gordon	181	150	31	12,533
Gravier	168	111	57	2905
Khan	63	23	40	2308
Nakayama	105	21	84	22,283
Pomeroy	60	39	21	7128
Shipp	77	58	19	7129
Singh	102	52	50	12,600
Sorlie	85	32	53	456
Subramanian	50	33	17	10,100
Sun	180	81	99	54,613
Tian	173	137	36	12,625
West	49	25	24	7129
Yeoh	248	79	169	12,625

WRAcc, k, time, number of tests and patterns obtained by SSDP and NMEEF-1k-1M algorithms in 10 microarray databases.

Base	Algorithm							
	NMEEF-1k-1M				SSDP			
	$WRAcc_{normalized}$	k	Time (s)	Tests (10^6)	$WRAcc_{normalized}$	k	Time (s)	Tests (10^6)
Alon	0.26	3	1984	1	0.572	5	0.422	0.116
Burczynski	0	0	63,697	1	0.684	5	8.254	1.247
Chiaretti	0	0	36,882	1	0.584	5	4.789	0.808
Chin	0.388	1	31,301	1	0.624	5	5.928	0.799
Christensen	0.592	1	3408	1	0.896	5	0.297	0.056
Gravier	0.232	1	5745	1	0.440	5	0.905	0.185
Nakayama	0	0	58,419	1	0.600	5	7.893	1.515
Tian	0	0	43,218	1	0.296	5	4.072	0.505
Yeoh	0	0	104,533	1	0.848	5	8.798	0.782
Sun	–	–	–	–	0.186	5	36.315	3.386

SSDP+

- Embora as medidas implantadas garantiam certo nível de eliminação de redundância, o que foi observado na prática é que a cobertura global ainda poderia ser melhorada caso outras medidas fossem implementadas
- Essas medidas, além de controlar a redundância, permitiam que padrões alternativos fossem encontrados e apresentados ao usuário. Esses padrões, por sua vez, podem ser mais úteis ao usuário
- Assim, Lucas, Vimieiro e Ludermir (2018) apresentaram uma extensão do SSDP para incluir outras formas de controle de redundância

SSDP+

- O algoritmo SSDP+ introduziu o conceito de cache para os padrões
- O conjunto Pk dos k melhores padrões passou a ser tratado como uma lista ordenada
- Ao preencher essa lista, o algoritmo verifica a similaridade de cobertura do novo padrão com respeito aos já inseridos, sequencialmente
- Caso o novo subgrupo s’ seja similar a algum subgrupo s já inserido, verifica-se se s’ possui qualidade inferior a s, se for o caso, então s’ é inserido no cache de s
  - Como o cache é limitado, ele deve substituir o de pior qualidade
- Se s’ for similar, mas possui maior qualidade, então ele assume o papel de s, e o cache é recriado (os elementos são reinserido em Pk)
- Se s’ não é similar a nenhum dos subgrupos já inseridos em Pk, ele é inserido à lista se o número de elementos presente for menor que k, e descartado caso contrário

SSDP+

- $item_{dom} = \max\{count(i)\}/k$
  - $count(i) = |\{X \in P_k | i \in X\}|$
- $SUPP^+ = \left|\bigcup_{X \in P_k} c^+(X)\right|/N$

Algorithm	Qg	$item_{dom}$	$SUPP^+$	time
Bioinformatics				
SSDP+s10	9.63	0.11	0.99	8.80
SSDP+s50	25.92	0.19	0.97	8.79
SSDP+s90	32.29	0.42	0.92	9.52
SSDP	32.07	0.38	0.93	6.61
SD	27.33	0.38	0.90	9.16
SD-RSS	27.06	0.29	0.92	9.04
Text mining				
SSDP+s10	7.09	0.10	0.65	26.24
SSDP+s50	8.55	0.16	0.61	25.19
SSDP+s90	12.42	0.46	0.39	30.06
SSDP	11.10	0.27	0.31	12.36
SD	18.02	0.65	0.36	55.77
SD-RSS	16.70	0.51	0.42	56.71
Humanities/social sciences				
SSDP+s10	15.463	0.37	0.05	536.62
SSDP+s50	21.355	0.46	0.04	541.76
SSDP+s90	29.268	0.59	0.02	516.16
SSDP	28.286	0.48	0.03	381.82
SD	17.842	0.99	0.03	716.78
SD-RSS	17.131	0.96	0.04	752.98

Leitura

- T. Lucas, R. Vimieiro and T. Ludermir, "SSDP+: A Diverse and More Informative Subgroup Discovery Approach for High Dimensional Data," *2018 IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, 2018, pp. 1-8, doi: 10.1109/CEC.2018.8477855.
- Tarcísio Lucas, Túlio C.P.B. Silva, Renato Vimieiro, and Teresa B. Ludermir. 2017. A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data. Appl. Soft Comput. 59, C (October 2017), 487–499. https://doi.org/10.1016/j.asoc.2017.05.048
- T. Pontes, R. Vimieiro and T. B. Ludermir, "SSDP: A Simple Evolutionary Approach for Top-K Discriminative Patterns in High Dimensional Databases," 2016 5th Brazilian Conference on Intelligent Systems (BRACIS), Recife, Brazil, 2016, pp. 361-366, https://doi.org/10.1109/BRACIS.2016.072

Aprendizado Descritivo

Aula 12 – Algoritmos evolucionários para descoberta de subgrupos

Professor Renato Vimieiro

DCC/ICEx/UFMG