Mineração de Dados: Trabalho Prático 1

Artur Rodrigues

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

artur@dcc.ufmg.br

1. INTRODUÇÃO

Técnicas de mineração de dados são amplamente utilizadas em diversos campos de aplicação como bancário, *marketing* e varejo. A mineração de padrões frequentes é uma técnica utilizada em mineração de dados para a descoberta de associações, aparentemente escondidas, que surgem entre vários itens [Agrawal et al. 1993]. No varejo, a análise da cesta de compras existe para descobrir quais itens geralmente são comprados em conjunto com o intuito de identificar padrões de compra dos consumidores e melhorar os negócios. No geral, aqueles que realizam a mineração de dados estão em busca de padrões frequentes de compras. Adicionalmente, têm se dado muita atenção aos padrões que são infrequentes ou excepcionais, como transações de cartões de crédito fraudulentas ou sintomas raros que implicam em doenças. Ainda na análise de uma cesta de compras, alguns conjuntos de itens, como arroz e feijão, ocorrem frequentemente e são associados como casos comuns. Em contraste, outros itens como carne de cordeiro e hortelã formam conjuntos de itens associados infrequentes, mas ainda sim relevantes. Além disso, outras associações podem ser encontradas que não eram previstas [Sadhasivam and Angamuthu 2011].

2. APRIORI

Consideramos $I = \{i_1, i_2, ..., i_m\}$ como um conjunto de itens. Seja T um conjunto de transações (possivelmente uma base de dados), onde cada transação t é um conjunto de itens tal que $t \subseteq I$. Uma *regra de associação* é uma implicação da forma $X \rightarrow Y$, onde $X \subset I$, $Y \subset I$ e $X \cap Y = \emptyset$. A transação $X \rightarrow Y$ se aplica ao conjunto de transações T com *confiança* c se $c\%$ das transações em T que "suportam" X também "suportam" Y. A regra tem *suporte* s em T se $s\%$ das transações em T contêm $X \cup Y$ [Liu et al. 1999].

Dado um conjunto de transações T, o problema de minera regras de associações é o de descobrir todas as regras de associação que possuem suporte e confiança maior que um valor mínimo especificado para suporte (*minsup*) e confiança (*minconf*).

Um algoritmo de mineração de associações trabalha, basicamente, em dois estágios:

1. geração de todos os conjuntos de itens que satisfazem *minsup*
2. geração de todas as regras de associação que satisfazem *minconf* usando os conjuntos gerados no passo anterior

O estágio 1 começa com a geração dos conjuntos de somente 1 item, procedimento que, na realidade, representa uma contagem do número de transações onde cada item i aparece. Em seguida, são gerados os conjuntos de 2 itens tomando pares a partir dos conjuntos de 1 item. Através desse mesmo procedimento são gerados conjuntos de k itens de maneira construtiva, até que seja atingido um limite para $k = |I|$.

Assim que todos os conjuntos de itens com suporte mínimo foram gerados o segundo estágio simplesmente envolve transformar cada um desses conjuntos em uma regra ou um conjunto de regras, que respeitem a confiança mínima.

2.1. Geração de conjuntos de itens de maneira eficiente

É importante notar que a métrica de *suporte* satisfaz a propriedade do fechamento. Se um conjunto de itens satisfaz um valor para *minsup* então todos seus subconjuntos também o satisfazem. Essa propriedade permite que a geração de conjuntos de itens seja feita de maneira mais eficiente.

No processo de geração de conjuntos de itens que satisfazem um valor de *minsup* são baseados num procedimento conhecido como busca *level-wise*. Consideremos k-itemset como um conjunto de itens com k itens que é "largo" quando tem suporte superior a *minsup*. Primeiramente são gerados todos os 1-itemsets largos, seguidos por todos os 2-itemsets e assim por diante. Mas é importante notar que se um itemset não é largo no nível $k-1$, ele é descartado já que qualquer adição de itens a esse conjunto não formará um conjunto largo (propriedade do fechamento). Assim, todos os potenciais itemsets largos no nível k são gerados a partir de itemsets largos no nível $k-1$ [Liu et al. 1999].

Um exemplo explica melhor esse procedimento. Supondo cinco 3-itemsets: $(ABC), (ABD), (ACD), (ACE)$, e (BCD). A união dos dois primeiros, $(ABCD)$ é um 4-itemset candidato porque os seus outros subconjuntos 3-itemsets (ACD) e (BCD) possuem suporte acima do mínimo. Se os 3-itemsets estão ordenados por ordem lexicográfica, como estão nesse exemplo, então é necessário considerar apenas pares com os mesmos dois primeiros membros. Por exemplo, não consideramos (ACD) e (BCD) porque $(ABCD)$ também pode ser gerado a partir de (ABC) e (ABD), e se esses dois não são 3-itemsets candidatos, então $(ABCD)$ não poderá ser um 4-itemset candidato. Isso nos deixa com os pares (ABC) e (ABD), que já foram explicados, e (ACD) e (ACE). Esse segundo par leva ao conjunto $(ACDE)$, cujos subconjuntos 3-itemsets não possuem todos o suporte mínimo, sendo assim descartado [Witten et al. 2011].

2.2. Geração de regras de maneira eficiente

Como mencionado anteriormente, o segundo estágio toma cada conjunto de itens e gera regras a partir deles, checando quais têm a confiança mínima. O maneira força bruta avalia o efeito de colocar cada subconjunto do lado direito da regra, chamado "consequente", deixando o restante o conjunto do lado esquerdo, chamado "antecedente". Naturalmente, esse método é caro computacionalmente a menos que os conjuntos sejam pequenos, porque o número de possíveis subconjuntos cresce exponencialmente com o tamanho dos conjuntos de itens.

Todavia, existe uma maneira mais inteligente. Se a regra com consequente duplo $(AB) \rightarrow (CD)$ possui suporte e confiança superiores ao mínimo estabelecido, as duas regras com consequentes únicos formadas a partir do mesmo conjunto de itens também respeitam esse valores mínimos: $(ABD) \rightarrow (C)$ e $(ABC) \rightarrow (D)$.

Reciprocamente, se uma das regras com consequentes únicos não possui os valores mínimos para confiança e suporte, não há razão para considerar a regra com consequente duplo. Através desse mecanismo é possível construir a partir de regras com consequentes únicos, novas com consequentes duplos, e a partir dessas, construir regras com consequentes triplos e assim em diante. Naturalmente, cada regra candidata deve ser testada para constatar se realmente possui a confiança mínima estabelecida [Witten et al. 2011].

2.3. Complexidade

A complexidade do algoritmo Apriori no pior caso é $(|I| \cdot |D| \cdot 2^{|I|})$, onde $I = \{i_1, i_2, ..., i_m\}$ é o conjunto de itens e D é o dicionário que contêm a frequência de cada subconjunto de itens de I.

Essa complexidade é equivalente ao do algoritmo força-bruta, já que todos os conjuntos de itens podem atender o mínimo estabelecido para a confiança. Na prática isso não ocorre, e o cálculo para essas situações foge do escopo desse trabalho. Mais detalhes podem ser encontrados em [Zaki and Junior 2012].

3. PROBLEMA DO ITEM RARO

O elemento chave que torna a mineração de regras de associação uma atividade prática é o valor mínimo para o suporte. Ele é usado para podar o espaço de busca e limitar o número de regras geradas. Todavia, a utilização de um valor único para *minsup* implicitamente assume que todos os itens nas transações têm a mesma natureza e/ou possuem frequências similares no banco de dados. Em aplicações da vida real, esse geralmente não é o caso, pois alguns itens aparecem raramente nas transações, enquanto outros aparecem muito frequentemente, ocasionando dois problemas:

1. Se *minsup* é definido muito alto, não serão encontradas regras que envolvem itens infrequentes ou raros nas transações;
2. Para que sejam geradas regras que envolvam tanto os itens frequentes quanto os raros, o valor de *minsup* deve ser definido muito baixo. Todavia, isso pode levar a uma explosão combinatorial, produzindo muitas regras, uma vez que os itens frequentes serão associados entre si de todas as maneiras possíveis, sendo muitas delas irrelevantes ou sem significado.

Esse dilema é conhecido como o *problema do item raro* [Liu et al. 1999]. O trabalho em questão utiliza uma proposta desenvolvida pelos alunos Artur Oliveira Rodrigues e Thales Filizola Costa e foi batizada de *Artur-Thales Metric* ou *atm*. Ela é uma medida para o um conjunto de itens $I = \{i_1, i_2, ..., i_m\}$, sendo definida como:

$$\Psi = \frac{freq(I)^m}{freq(i_1)freq(i_2)...freq(i_m)}$$

É importante notar que o intervalo para *atm* é $[0, 1]$, e que assim como a confiança e o suporte, ele admite um valor mínimo. Essa nova medida só é utilizada quando é identificado um conjunto de itens com suporte abaixo do mínimo estabelicido (um possível conjunto com itens raros), quando será calculado de *atm* para esse conjunto. Somente caso o valor seja inferior ao mínimo estabelecido que o conjunto de itens será descartado. O pseudo-código abaixo ilustra essa abordagem:

Algorithm 1: Remoção de conjunto de item

```
if sup do itemset < minsup then
    if atm do itemset < minatm then
        remove itemset
    end
end
```

Vale ressaltar que quando essa medida é utilizada, todos os 1-itemsets são gerados, pois somente assim será possível calcular o denominador da métrica *atm*. Caso não seja fornecido um valor mínimo para *atm* o algoritmo tradicional para o *Apriori* é utilizado.

3.1. Implementação

O algoritmo foi implementado utilizando a linguagem *Python*, pois essa provê algumas estruturas de dados como conjuntos que facilitam o desenvolvimento do algoritmo.

4. BASE DE DADOS

A base de dados utilizada foi disponibilizada pela comissão avaliadora da disciplina, apresentando dados de uma pesquisa semelhante a um censo populacional, possuindo 48842 transações do tipo atributo-valor, como por exemplo *age=senior*. Conclui-se que em uma transação não existiram dois valores diferentes para um mesmo atributo, já que isso não deve ocorrer em nenhuma circunstância.

É importante notar que a base disponibilizada foi modificada de uma maneira que a tornou inconsistente, pois as transações que possuem os atributos *gain* e *loss* deveriam ter sido eliminadas, o que não foi feito. Dada a ausência de significado para esses atributos, a existência deles na base pode levar a geração de regras desinteressantes. As quinze 1-itemsets mais frequentes estão exibidos na figura 1.

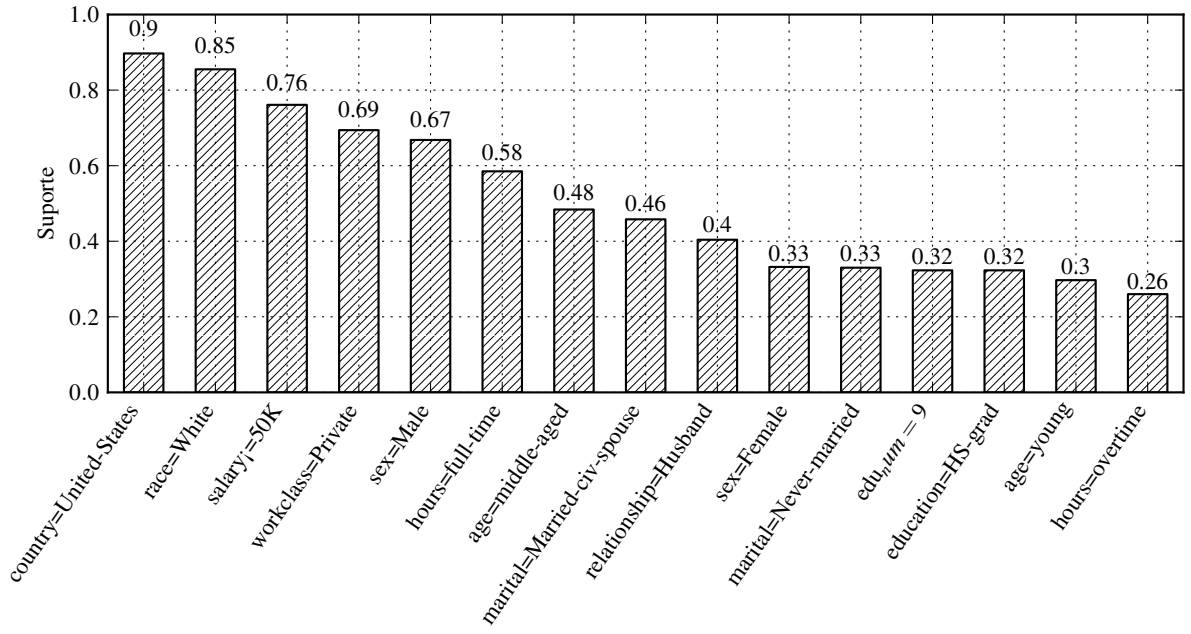


Figura 1: Os 15 1-itemsets mais frequentes

5. AVALIAÇÃO EXPERIMENTAL

5.1. Procedimentos

Com o intuito de se obter testes mais consistentes, os experimentos foram executados em ambiente virtualizado, com capacidade de processamento e memória primária reduzidas, 50% da capacidade da máquina hospedeira e 1024MiB, respectivamente. O sistema operacional do ambiente virtualizado era Ubuntu Server 12.04 64 bits e os softwares utilizados foram interpretador Python (2.7.2) PyPy versão 1.9.0, e GCC versão 4.2.1. A máquina hospedeira possuía sistema operacional Mac OS X 10.8.2, processador *quad-core* de 2.3GHz e memória primária com capacidade de 16GiB.

Todos os testes foram realizados 5 vezes e o resultado médio para o tempo de execução foi considerado. Finalmente, certificou-se que a solução desenvolvida execute perfeitamente na estação `claro.grad.dcc.ufmg.br`.

5.2. Análise de Parâmetros

Os parâmetros *minsup* e *minconf* foram variados no intervalo $[0.3, 0.9]$ com saltos de 0.1, os resultados podem ser vistos na figura 2. Como é de se esperar, o número de regras é maior para valores menores dos dois parâmetros. Para *minconf* e *minsup* com valor 0.3 foram geradas 416 regras, sendo que muitas dessas regras são insignificantes como, por exemplo

```
country=United-States,marital=Married-civ-spouse,race=White,
    relationship=Husband -> sex=Male 1.000
```

uma vez que se o indivíduo é "husband" (marido), obviamente ele é "male" (do sexo masculino), justificando o valor de confiança 1.000. É importante observar entretanto que se esse itens, *relationship=Husband* e *sex=Male* fossem altamente frequentes, essa regra ainda seria encontrada para valores altos dos parâmetros. Outra regra pouco interessante é

```
country=United-States -> age=middle-aged,sex=Male 0.338
```

por se tratar de uma regra que reflete um viés desinteressante da base de dados. Um aumento dos valores desses parâmetros para 0.5 implica na geração de regras mais relevantes, como

```
salary<=50K -> workclass=Private 0.714
```

que claramente traz uma informação interessante, a de que se o salário de um indivíduo é inferior a 50 mil, ele tem 71.4% de chances de estar empregado no setor Privado.

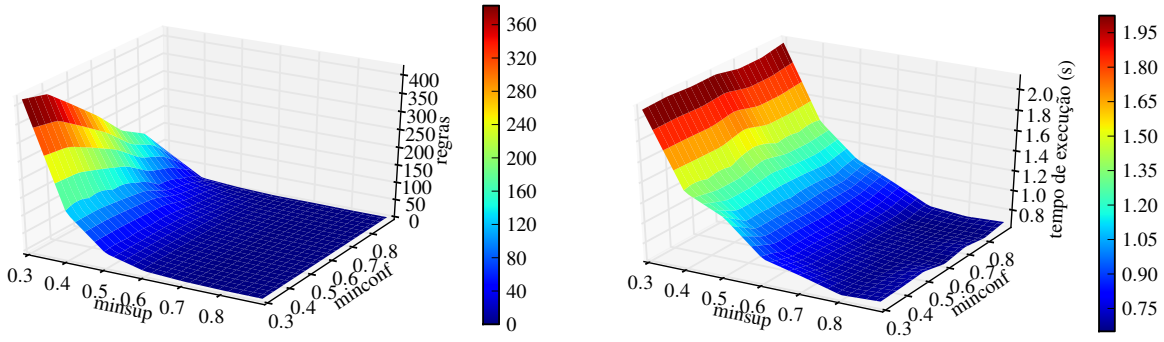
Finalmente, ainda pelo gráfico da figura 2a, é possível notar que o aumento tanto no valor de *minsup* quanto no de *minconf* implica numa diminuição do número de regras geradas, com um declínio um pouco mais acentuado para o primeiro parâmetro. Valores superiores a 0.8 para *minsup* já implicam na nulidade de regras geradas.

É interessante notar na figura 2b que o tempo de execução do algoritmo diminui em ordem logarítmica com o aumento do valor de *minsup*, enquanto um aumento no valor de *minconf* não implica em alterações evidentes no tempo de execução.

5.3. Análise da Qualidade da Solução

Inicialmente, uma análise da solução com vistas somente no algoritmo Apriori não é relevante para essa documentação, uma vez que não foram disponibilizadas informações complementares da base de dados que permitissem um estudo comparativo e qualitativo das regras gerados. Assim, essa seção focará na análise das implicações da proposta de solução para o problema do item raro.

Da mesma maneira que para o suporte, quanto menor o valor do *atm*, mais conjuntos de itens serão gerados, causando um aumento no custo computacional. Em específico, a solução proposta, Ψ, envolve a busca de $|I| + 1$ valores de frequências para cada conjunto de itens $I = \{i_1, i_2, ..., i_m\}$, enquanto o cálculo do suporte envolve a busca de somente um valor. Tendo sido estabelecidos os valores $minsup = 0.5$ e $minconf = 0.7$, variou-se o valor de *minatm* no



(a) Regras (b) Tempo de execução

Figura 2: Impacto dos valores para *minsup* e *minconf*

intervalo $[0.2, 1.0]$, com saltos de 0.1. A influência no tempo de execução e no número de regras geradas pode ser visto na figura 3. Fica claro que com o aumento da restrição para *atm* menos regras são produzidas. Por outro lado, o tempo de execução não sofre variação significante. Isso se deve ao fato de que a o impacto da inclusão de novos conjuntos de itens é insignificante perto do custo de se testar cada conjunto de itens para o valor de Ψ. Vale ainda ressaltar que o tempo de execução sem o *atm*, para esses valores de suporte e confiança é de 1.019 segundos.

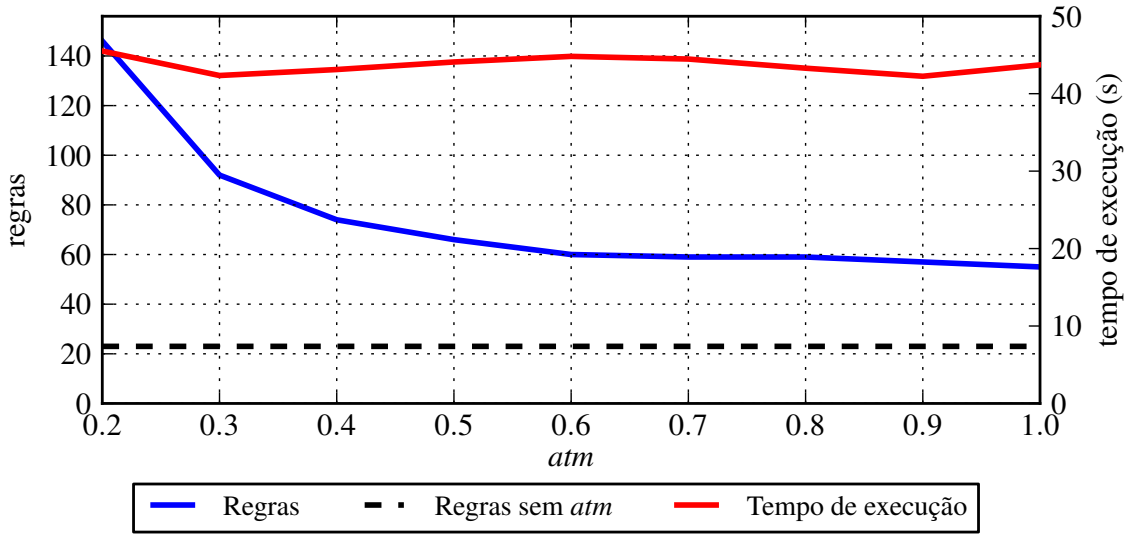


Figura 3: Impacto de *atm* no número de regras geradas e no tempo de execução

Se observarmos as regras geradas quando o valor de $minatm = 1.0$, que significa incluir conjuntos de itens infrequentes mas cujos itens sempre co-ocorrem juntos, observamos que novas regras são identificadas (regras que não foram geradas com os mesmos valores de *minsup* e *minconf* no Apriori tradicional), como por exemplo

```
edu_num=2 -> education=1st-4th 1.000
edu_num=3 -> education=5th-6th 1.000
edu_num=4 -> education=7th-8th 1.000
```

que estão mostrando atributos redundantes na base de dados. Se a base fosse de cesto de compras, esse tipo de regra poderia evidenciar itens raros mas que sempre são comprados juntos, como por exemplo, esmalte e acetona. A geração dessas regras corrobora a alta qualidade da solução para o problema do item raro apresentada. Adicionalmente, o suporte para o item `edu_num=3`, por exemplo, é 0.01 o que faz dele um item raro.

Se relaxarmos o valor de *minatm* para 0.8, novas regras são geradas, como por exemplo

```
age=young -> marital=Never-married 0.744
salary>50K -> relationship=Husband 0.757
hours=full-time -> workclass=Private 0.722
```

que novamente, são regras que possuem itens raros, mas que são extremamente interessantes, novamente corroborando a qualidade da solução desenvolvida nesse trabalho.

6. CONCLUSÃO

Nesse trabalho foi apresentado um estudo sobre a mineração de conjuntos de itens frequentes, em especial com vistas no algoritmo Apriori. O problema do item raro foi caracterizado e uma solução foi proposta para contornar esse problema. Essa solução se mostrou boa ao identificar regras que envolvem itens raros, mas que ainda sim são relevantes para o estudo proposto. Além disso, foi avaliado experimentalmente o impacto dos valores para confiança e suporte mínimo.

Referências

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. ACM Request Permissions.

Liu, B., Hsu, W., and Ma, Y. (1999). Mining association rules with multiple minimum supports. pages 337–341.

Rawat, S. S. and Rajamani, L. (2011). Probability apriori based approach to mine rare association rules. In *Data Mining and Optimization (DMO), 2011 3rd Conference on*, pages 253–258.

Sadhasivam, K. and Angamuthu, T. (2011). Mining Rare Itemset with Automated Support Thresholds. *Journal of Computer Science*.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *Sigkdd Explorations*, 6(1):7–19.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition.

Zaki, M. and Junior, W. M. (2012). *Fundamentals of Data Mining Algorithms*. Cambridge, draft edition.