Regras de Associação

ESTAT0109 – Mineração de Dados em Estatística

Prof. Dr. Sadraque E. F. Lucena

sadraquelucena@academico.ufs.br

http://sadraquelucena.github.io/mineracao

Objetivo da Aula

- Compreender o problema da Análise de Cesta de Mercado e o conceito de Regras de Associação no formato SE (Antecedente) → ENTÃO (Consequente).
- Dominar as métricas fundamentais (Suporte, Confiança e Lift) para filtrar e avaliar a força e o interesse dos padrões de afinidade entre itens.
- Entender o funcionamento do Algoritmo Apriori e a estratégia FP-Growth para a mineração eficiente de itemsets frequentes em grandes volumes de dados de transações.

O Problema: Análise de Cesta de Mercado

As Regras de Associação nasceram da necessidade de entender o comportamento de consumo. O problema clássico é a **Análise de Cesta de Mercado** (*Market Basket Analysis*).

- **Pergunta:** Quais itens os clientes tendem a comprar *juntos*?
- **Objetivo:** Identificar padrões de afinidade para otimizar *layout* de loja, criar promoções (ex: “compre X e Y, ganhe desconto”), e planejar o inventário.

Definições Formais

- **Item:** Um produto (ex: Pão, Leite, Fralda).
- **Transação:** Uma única “cesta de compras” (ex: T1, T2, …).
- **Itemset (Conjunto de Itens):** Uma coleção de 1 ou mais itens (ex: {Leite, Fralda}).

O Problema: Análise de Cesta de Mercado

Exemplo Base (N=5 Transações)

Vamos usar esta base de dados para todos os nossos cálculos:

Transação (TID)	Pão	Leite	Cerveja	Fralda	Ovos	Refrigerante
T1	1	1	1	0	0	0
T2	1	0	1	1	1	0
T3	0	1	1	1	0	1
T4	1	1	1	1	0	0
T5	1	1	0	1	0	1
Total (N=5)	**4**	**4**	**3**	**4**	**1**	**2**

A Regra

Uma Regra de Associação tem o formato **SE** $\rightarrow$ **ENTÃO**:

$$\text{Antecedente} \rightarrow \text{Consequente}$$

$$\{X\} \rightarrow \{Y\}$$

- O **Antecedente (X)** é um *itemset*.
- O **Consequente (Y)** é um *itemset*.
- **Restrição:** X e Y são disjuntos ($X \cap Y = \emptyset$).
- **Exemplo:** A regra $\{\text{Cerveja}, \text{Fralda}\} \rightarrow \{\text{Pão}\}$ sugere que clientes que compram Cerveja e Fralda também tendem a comprar Pão.

A Regra

O Desafio Computacional

Com p itens distintos, o número total de regras possíveis é $3^p - 2^{p+1} + 1$.

- Para 6 itens (nosso exemplo): $3^6 - 2^7 + 1 = 602$ regras.
- Para 50 itens (mercearia pequena): 1.125×10^{15} (mais de 1 quatrilhão) de regras.

Conclusão: Não podemos avaliar todas. Precisamos de métricas para **filtrar** e encontrar apenas as regras “fortes” ou “interessantes”.

Métricas Fundamentais

Para uma regra $\{X\} \rightarrow \{Y\}$, usamos três métricas principais.

1. Suporte (Relevância)

O Suporte mede a **frequência** ou **popularidade** de um *itemset* no banco de dados.

$$Suporte(X) = P(X) = \frac{\text{Nº de transações contendo o itemset } X}{\text{Total de transações (N)}}$$

- **Suporte da Regra:** O suporte de uma regra $\{X\} \rightarrow \{Y\}$ é o suporte do *itemset* que contém *ambos* os lados.

$$Suporte(X \rightarrow Y) = Suporte(X \cup Y) = P(X \cap Y)$$

Métricas Fundamentais

1. Suporte (Relevância)

Usando nossa tabela (N=5):

a. **Suporte de 1 item:**

- $Suporte(\{\text{Pão}\}) = 4/5 = 0.8$
- $Suporte(\{\text{Leite}\}) = 4/5 = 0.8$
- $Suporte(\{\text{Cerveja}\}) = 4/5 = 0.8$
- $Suporte(\{\text{Fralda}\}) = 4/5 = 0.8$

b. **Suporte de *Itemset* (2 itens):**

- Itemset: $\{\text{Cerveja, Leite}\}$
- Ocorrências: T1, T3, T4 (3 transações)
- $Suporte(\{\text{Cerveja, Leite}\}) = 3/5 = 0.6$

Métricas Fundamentais

1. Suporte (Relevância)

Usando nossa tabela (N=5):

c. **Suporte de *Itemset* (3 itens):**

- Itemset: $\{\text{Cerveja, Leite, Fralda}\}$
- Ocorrências: T3, T4 (2 transações)
- $Suporte(\{\text{Cerveja, Leite, Fralda}\}) = 2/5 = 0.4$

Métricas Fundamentais

2. Confiança (Precisão)

A Confiança mede a **probabilidade** de o Consequente (Y) aparecer, *dado que* o Antecedente (X) já apareceu. É uma medida de probabilidade condicional, $P(Y|X)$.

$$Confiança(X \rightarrow Y) = P(Y|X) = \frac{Suporte(X \cup Y)}{Suporte(X)}$$

- **Interpretação:** “De todas as vezes que X foi comprado, em qual percentual Y também foi comprado?”

Métricas Fundamentais

2. Confiança (Precisão)

Cálculo (Regra: $\{\text{Cerveja, Leite}\} \rightarrow \{\text{Fralda}\}$**)**

$$Confiança(\{\text{Cerveja, Leite}\} \rightarrow \{\text{Fralda}\}) = \frac{Suporte(\{\text{Cerveja, Leite, Fralda}\})}{Suporte(\{\text{Cerveja, Leite}\})}$$

$$= \frac{2/5}{3/5} = 0.67$$

- **Interpretação:** “67% dos clientes que compraram Cerveja e Leite também compraram Fralda.”

Métricas Fundamentais

2. Confiança (Precisão)

- A Confiança pode ser enganosa.
- Imagine a regra $\{\text{Qualquer Coisa}\} \rightarrow \{\text{Pão}\}$.
- Se “Pão” for um item extremamente popular (ex: $Suporte(\{\text{Pão}\}) = 95\%$), quase *qualquer* regra apontando para “Pão” terá uma Confiança alta.
- Isso **não** significa que o Antecedente “causa” a compra do Pão; significa apenas que o Pão é comprado o tempo todo, independentemente.
- **Precisamos de uma métrica que desconte a popularidade do Consequente.**

Métricas Fundamentais

3. Lift (Interesse / “Poder de Tração”)

O Lift mede o quão mais (ou menos) provável é que X e Y ocorram *juntos* do que se fossem estatisticamente independentes. É o “teste” de interesse da regra.

$$Lift(X \rightarrow Y) = \frac{Confiança(X \rightarrow Y)}{Suporte(Y)} = \frac{P(Y|X)}{P(Y)}$$

Forma alternativa (e simétrica):

$$Lift(X \rightarrow Y) = \frac{P(X \cup Y)}{P(X)P(Y)} = \frac{Suporte(X \cup Y)}{Suporte(X) \times Suporte(Y)}$$

Métricas Fundamentais

3. Lift (Interesse / “Poder de Tração”)

- **Interpretação do Lift:**
  - **Lift = 1:** Independência. A ocorrência de X não altera a probabilidade de Y. A regra é inútil.
  - **Lift > 1:** Associação positiva. X e Y aparecem juntos mais do que o esperado. X “puxa” Y. (Interessante!)
  - **Lift < 1:** Associação negativa. X e Y aparecem juntos menos do que o esperado. X “inibe” Y. (Também interessante!)

Métricas Fundamentais

3. Lift (Interesse / “Poder de Tração”)

Cálculo (Regra: $\{\text{Cerveja, Leite}\} \rightarrow \{\text{Fralda}\}$**)**

- $Confiança(X \rightarrow Y) = 2/3$
- $Suporte(Y) = Suporte(\{\text{Fralda}\}) = 4/5$
- $Lift = (2/3)/(4/5) = (2/3) \times (5/4) = 10/12 \approx 0.833$
- **Interpretação:** O Lift é **menor que 1**. Isso significa que clientes que compram {Cerveja, Leite} são, na verdade, *um pouco menos prováveis* (cerca de 17% menos prováveis) de comprar Fralda do que um cliente aleatório.
- **Conclusão da Regra:** Embora a Confiança de 67% parecesse alta, o Lift nos mostra que esta regra **não é acionável** e a associação é, na verdade, negativa.

Métricas Adicionais

Para uma análise de nível sênior, Lift, Suporte e Confiança são o mínimo. Duas outras métricas dão uma visão mais completa:

4. **Leverage (Alavancagem):**

- $Leverage(X \rightarrow Y) = Suporte(X \cup Y) - (Suporte(X) \times Suporte(Y))$
- **O que mede:** A diferença *absoluta* entre a frequência observada de (X e Y) e a frequência esperada se fossem independentes.
- **Interpretação:** Um valor de 0 indica independência. Um valor positivo indica quantos *mais* transações (em proporção) contêm X e Y do que o esperado. É útil para medir o impacto em números absolutos, não apenas relativos.

Métricas Adicionais

5. **Conviction (Convicção):**

- $Conviction(X \rightarrow Y) = \frac{1 - Suporte(Y)}{1 - Confiança(X \rightarrow Y)}$
- **O que mede:** O grau de “erro” que a regra faria se a associação não existisse. Mede o quão dependente o Consequente é do Antecedente.
- **Interpretação:** Uma Convicção alta significa que o Consequente (Y) raramente aparece *sem* o Antecedente (X). É uma medida de direcionalidade muito mais forte que o Lift.

Classificando as Regras (O Que Fazer)

O objetivo não é apenas encontrar regras, mas classificá-las quanto à utilidade:

- **Acionáveis:** Regras com bom Suporte, Confiança e Lift > 1. Elas fornecem *insights* claros que podem ser aplicados (ex: “Coloque o {Achocolatado} perto do {Leite Condensado}”).
- **Triviais:** Regras que são óbvias para qualquer especialista no domínio (ex: $\{\text{Caneta}\} \rightarrow \{\text{Caderno}\}$). Elas validam o modelo, mas não geram *insights* novos.
- **Inexplicáveis:** Regras que desafiam a lógica (ex: $\{\text{Sapatos}\} \rightarrow \{\text{Canetas}\}$). Podem ser ruído estatístico (baixo Suporte) ou exigir mais pesquisa para serem compreendidas.

O Algoritmo Apriori (Como Fazer)

- Como evitamos o 1 quatrilhão de cálculos? Usamos o **Algoritmo Apriori**.
- O *insight* genial do Apriori é baseado em uma propriedade chamada **“Fechamento para Baixo”** (Downward Closure Property).

Ideia do algoritmo

1. **Se um *itemset* é frequente (passa no limiar de Suporte), todos os seus subconjuntos também devem ser frequentes.** *(Isso é óbvio: se {Pão, Leite} é frequente, {Pão} tem que ser* pelo menos* tão frequente).*
2. **COROLÁRIO (A Poda): Se um *itemset* é INFREQUENTE (falha no Suporte), todos os seus SUPERCONJUNTOS também serão infrequentes.** *(Se {Ovos} é infrequente, não precisamos nem* calcular* o suporte de {Ovos, Pão} ou {Ovos, Pão, Leite}. Sabemos que eles falharão).*

O Algoritmo Apriori

Etapas do Algoritmo (Join & Prune)

O Apriori funciona “de baixo para cima”, construindo *itemsets* maiores a partir dos menores.

O Algoritmo Apriori

Etapas do Algoritmo (Join & Prune)

3. **Passo k:**
    - **Join:** Gere candidatos C(k) a partir dos frequentes L(k-1).
    - **Poda (Apriori):** Verifique se *todos* os subconjuntos de (k-1) itens de um candidato C(k) estão em L(k-1). Se não, pode o candidato.
    - **Poda (Suporte):** Calcule o Suporte de C(k) e descarte os infrequentes.
4. **Fim:** O processo para quando não há mais *itemsets* frequentes a serem gerados.

Uma Alternativa Eficiente: FP-Growth

- Embora o Apriori seja o algoritmo clássico, ele possui um gargalo computacional severo.

O Gargalo do Apriori

- O problema do Apriori não é o cálculo do suporte. O problema é a **geração de candidatos**.
  1. **Explosão Combinatória:** O processo de "Join" (juntar L(k-1) para criar C(k)) pode gerar um número astronômico de candidatos que precisarão ter seu suporte contado.
  2. **Múltiplas Varreduras:** O Apriori precisa varrer o banco de dados inteiro k vezes (uma vez para cada nível de itemset). Se $k = 10$, são 10 varreduras.
- Para bancos de dados muito grandes ou com padrões longos (ex: compras complexas, sequenciamento de DNA), o Apriori se torna computacionalmente inviável.

A Solução FP-Growth: Comprimir e Dividir

- O algoritmo **FP-Growth** (Frequent Pattern Growth) ataca esses dois gargalos:
  1. **Sem Geração de Candidatos:** O FP-Growth não gera itemsets candidatos.
  2. **Apenas 2 Varreduras:** Ele varre o banco de dados **apenas duas vezes**, independentemente do número de itens.
- **A Grande Ideia:** Em vez de varrer o banco de dados repetidamente para testar os itemsets, o FP-Growth comprime o banco de dados inteiro em uma estrutura de dados em árvore (a **FP-Tree**) e minera essa árvore compacta diretamente na memória.

A Solução FP-Growth: Comprimir e Dividir

Etapa 1: Construindo a FP-Tree (Frequent Pattern Tree)

1. **Varredura 1 (Scan 1):**
   - Conta o suporte de todos os itens individuais (1-itemsets).
   - Descarta os itens infrequentes (abaixo do min_suporte).
2. **Ordenação (Sorting):**
   - Cria uma “lista de itens frequentes” (F-List) ordenada por suporte (do mais frequente para o menos frequente).
   - **Por quê?** Isso garante que os itens mais comuns fiquem mais próximos da raiz da árvore, maximizando a compressão.

A Solução FP-Growth: Comprimir e Dividir

Etapa 1: Construindo a FP-Tree (Frequent Pattern Tree)

3. **Varredura 2 (Scan 2):**
    - Lê cada transação, uma por uma.
    - Filtra apenas os itens frequentes daquela transação.
    - Ordena esses itens de acordo com a F-List.
    - Insere a transação ordenada na **FP-Tree**.

A Solução FP-Growth: Comprimir e Dividir

Etapa 2: Mineração da FP-Tree (Divide and Conquer)

FP-Growth usa uma estratégia de “Dividir e Conquistar”.

1. Começa pela Tabela de Cabeçalho, do item menos frequente (no final da lista).
2. Para esse item (ex: “Ovos”), ele coleta todos os “caminhos prefixos” que terminam em “Ovos”. Isso forma sua Base de Padrões Condicionais.
3. A partir dessa base, ele constrói uma nova FP-Tree Condicional (uma árvore muito menor, apenas para “Ovos”).
4. Ele minera recursivamente essa pequena árvore.
5. Todos os padrões encontrados (ex: {Pão}, {Pão, Leite}) são combinados com o item original (ex: {Pão, Ovos}, {Pão, Leite, Ovos}).

- Ao quebrar o problema grande (minerar a árvore inteira) em problemas menores (minerar sub-árvores condicionais), o FP-Growth é exponencialmente mais rápido que o Apriori em datasets densos.

Apriori vs. FP-Growth

Característica	Apriori	FP-Growth
Abordagem	“Gerar e Testar”” (Join & Prune)	“Dividir e Conquistar” (Pattern Growth)
Geração de Candidatos	Sim (Principal gargalo)	Não
Varreduras no Banco	$k + 1$ varreduras (lento)	2 varreduras (rápido)
Uso de Memória	Baixo (só mantém L(k-1))	Alto (A FP-Tree precisa caber na memória)
Quando Usar?	*Datasets* esparsos, simples, ou quando k é pequeno.	*Datasets* densos, grandes, ou com padrões longos.

Agora vamos fazer no R...