Instituto Federal de Educação, Ciência e Tecnologia da Bahia

ANDREY GOMES DA SILVA NASCIMENTO
GABRIEL NASCIMENTO MIRANDA DOS SANTOS

Relatório Técnico: Implementação de Regras de Associação

Salvador - BA
2026

1. Introdução

No contexto de Business Intelligence (BI) e Data Warehousing, a descoberta de padrões ocultos em grandes volumes de dados é essencial para a tomada de decisões estratégicas. As Regras de Associação permitem identificar correlações do tipo ”quem compra X, também compra Y”, fundamentais para estratégias de vendas cruzadas (cross-selling) e organização de layout de lojas.

Este relatório documenta a implementação de uma solução de mineração de dados que lê transações de um arquivo em disco, processa as informações e gera regras baseadas em limiares de Suporte e Confiança.

2. Escolha do Algoritmo: Apriori

Para a resolução do problema proposto, foi selecionado o algoritmo Apriori. Esta escolha baseia-se em três pilares principais:

- **Eficiência por Poda**: Diferente de algoritmos de força bruta que testam todas as combinações possíveis de itens, o Apriori utiliza a propriedade da monotonicidade. O princípio fundamental é: ”Se um conjunto de itens é frequente, então todos os seus subconjuntos também devem ser frequentes”. Inversamente, se um item é raro, qualquer combinação que o contenha também será rara e pode ser descartada imediatamente.
- **Adequação Acadêmica e Prática:** É o algoritmo clássico para ensino de Regras de Associação, sendo robusto o suficiente para bases de dados transacionais típicas de varejo.
- **Simplicidade de Implementação**: A lógica iterativa baseada em contagem de frequência e filtragem facilita a implementação em linguagens de alto nível sem a necessidade de bibliotecas complexas de terceiros.

3. Suporte e Confiança

O algoritmo opera calculando duas métricas principais para validar a força de uma associação entre itens A e B:

3.1 Suporte

Mede a frequência com que o conjunto de itens aparece na base de dados total.

$Sup(A \cup B)$ = Transações contendo A e B / Número Total de Transações

3.2 Confiança

Mede a probabilidade de se encontrar o item B, dado que o item A está presente.

$Conf(A \rightarrow B) = Sup(A \cup B) / Sup(A)$

4. Implementação

A solução foi desenvolvida em JavaScript (ambiente Node.js), aproveitando a manipulação nativa de objetos JSON e operações de conjunto.

4.1 Configuração e Leitura de Dados

O sistema define constantes para os limiares mínimos (MIN_SUPPORT = 0.2 e MIN_CONFIDENCE =0.3) e carrega os dados de um arquivo externo data.json, simulando um ambiente de produção onde os dados persistem em disco.

4.2 Passo 1: Identificação de Itens Frequentes

A primeira etapa do algoritmo consiste em varrer a base de dados para contar a ocorrência individual de cada item. Aplica-se imediatamente o filtro de suporte mínimo.

```
allItems.forEach( item => {
  const count = data.filter( t => t.items.includes(item) ).length;
  const support = count / totalTransactions;
  itemCount[item] = count;

  if(support >= MIN_SUPPORT){
    frequentItems.push(item);
  }
});
```

Esta etapa é crucial para a performance, pois elimina itens irrelevantes antes que o processamento de combinações comece.

4.3 Passo 2: Geração de Pares e Regras

Utilizando apenas os itens aprovados na etapa anterior, o algoritmo gera combinações em pares. Para cada par, verifica-se novamente o suporte conjunto e, se válido, calcula-se a confiança em ambas as direções (A → B e B → A).

```
const candidatePairs = getCombinations(frequentItems);

candidatePairs.forEach( pair => {
  // cálculo do suporte do parr
  const [itemA, itemB] = pair;
  const countAB = data.filter(t => transactionsContain(t, pair))
    .length;
  const supportAB = countAB / totalTransactions;
```

```
  if(supportAB >= MIN_SUPPORT){
    // confiança A->B
    const countA = itemCount[itemA];
    const confidenceAtoB = countAB / countA;

    if(confidenceAtoB >= MIN_CONFIDENCE){
      printRule(itemA, itemB, supportAB, confidenceAtoB);
    }

    // confiança B->A
    const countB = itemCount[itemB];
    const confidenceBtoA = countAB / countB;

    if(confidenceBtoA >= MIN_CONFIDENCE){
      printRule(itemB, itemA, supportAB, confidenceBtoA);
    }
  }
});
```

4.4 Saída

Utilizando como exemplo a base de dados fornecida pelo slide (***DW, SSP e Mineração de dados***) e a funções de I/O para visualizar a saída de dados, é retornada essas estatísticas:

```
Initializing (Min Support: 0.2 , Min Confidence: 0.3 )
Total transactions: 10
Frequent items: [
  'cafe',     'pao',
  'manteiga', 'leite',
  'cerveja',  'feijao',
  'arroz'
]
Regra: [cafe] => [pao]
   Suporte: 30.0% | Confiança: 100.0%
---
Regra: [pao] => [cafe]
   Suporte: 30.0% | Confiança: 60.0%
---
Regra: [cafe] => [manteiga]
   Suporte: 30.0% | Confiança: 100.0%
```

```
---
Regra: [manteiga] => [cafe]
   Suporte: 30.0% | Confiança: 60.0%
---
Regra: [pao] => [manteiga]
   Suporte: 40.0% | Confiança: 80.0%
---
Regra: [manteiga] => [pao]
   Suporte: 40.0% | Confiança: 80.0%
---
Regra: [pao] => [leite]
   Suporte: 20.0% | Confiança: 40.0%
---
Regra: [leite] => [pao]
   Suporte: 20.0% | Confiança: 100.0%
---
Regra: [manteiga] => [leite]
   Suporte: 20.0% | Confiança: 40.0%
---
Regra: [leite] => [manteiga]
   Suporte: 20.0% | Confiança: 100.0%
---
```

5. Conclusão

A implementação demonstrou com sucesso a aplicação prática dos conceitos de Mineração de Dados. O uso do algoritmo Apriori permitiu filtrar eficientemente o espaço de busca, focando apenas nos padrões estatisticamente relevantes.

O código desenvolvido é modular e respeita as restrições de desempenho ao evitar cálculos desnecessários para itens infrequentes. A solução é capaz de identificar associações fortes, fornecendo insights valiosos que poderiam ser utilizados, por exemplo, para recomendação de produtos ou otimização de inventário.