Artigo 4: "An Extensive Comparative Study of Cluster Validity Indices" (Arbelaitz et al., 2013)

---

1. Introdução

Os algoritmos de agrupamento (ou clustering) são amplamente utilizados em diversas áreas, como bioinformática, mineração de dados, aprendizado de máquina e visão computacional. Entretanto, não existe um algoritmo de clustering universalmente ótimo para todos os problemas. A escolha do número de clusters e a avaliação da qualidade das partições geradas por esses algoritmos são desafios centrais.

Este artigo apresenta uma análise comparativa abrangente de **índices de validade de cluster (Cluster Validity Indices, CVIs)**, explorando como diferentes índices internos podem ser utilizados para avaliar a qualidade das partições geradas. Os CVIs internos medem a qualidade de um agrupamento considerando a **coesão intra-cluster** (objetos dentro de um cluster são semelhantes) e a **separação inter-cluster** (os clusters são bem separados).

Os principais objetivos do artigo incluem:

1. Identificar quais índices apresentam desempenho consistente em diferentes cenários.
2. Analisar os pontos fortes e limitações dos CVIs internos.
3. Fornecer recomendações práticas para a escolha de índices em diferentes condições de agrupamento.

---

2. Objetivo do Estudo

O estudo se propõe a responder as seguintes questões:

- Quais CVIs são mais adequados para diferentes estruturas de dados, como clusters esféricos, alongados ou sobrepostos?
- Como os índices internos se comportam em presença de ruído e alta dimensionalidade?
- Existe um índice que apresente desempenho consistente em todos os cenários?

---

3. Metodologia

3.1 Dados Utilizados

1. **Conjuntos de Dados Sintéticos**:
    - Gerados com diferentes características, como forma dos clusters (esféricos, alongados) e níveis de sobreposição.
    - Variação nos parâmetros: número de clusters (KK), dimensionalidade, e ruído.
2. **Conjuntos de Dados Reais**:
    - Selecionados de repositórios como UCI Machine Learning Repository.
    - Exemplos: Iris, Wine, e Glass.

3.2 Algoritmos de Clustering

Os CVIs foram avaliados em partições geradas por três algoritmos de clustering:

- **k-means**: Otimiza a coesão intra-cluster, assumindo clusters esféricos.
- **Ward**: Baseado na redução da variância intra-cluster em cada fusão.
- **Agrupamento Hierárquico**: Utilizando critérios como ligação média (average-linkage).

3.3 Índices de Validade Interna Avaliados

Foram analisados 30 CVIs internos, incluindo:

- **Índice de Silhueta** (S(i)S(i)):

  S(i)=b(i)−a(i)max(a(i),b(i))S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}

  Onde:

  - a(i)a(i): Distância média de ii para os pontos do mesmo cluster.
  - b(i)b(i): Distância mínima de ii para pontos de outros clusters.
- **Índice de Dunn**:

  D=min1≤i<j≤kdist(Ci,Cj)max1≤kdiam(Ck)D = \frac{\min_{1 \leq i < j \leq k} \text{dist}(C_i, C_j)}{\max_{1 \leq k} \text{diam}(C_k)}

  Onde:

  - dist(Ci,Cj)\text{dist}(C_i, C_j): Distância mínima entre clusters CiC_i e CjC_j.
  - diam(Ck)\text{diam}(C_k): Diâmetro do cluster CkC_k.
- **Índice de Davies-Bouldin (DB)**:

  DB=1K∑i=1Kmaxj≠iσi+σjdist(Ci,Cj)DB = \frac{1}{K} \sum_{i=1}^K \max_{j \neq i} \frac{\sigma_i + \sigma_j}{\text{dist}(C_i, C_j)}

  Onde:

  - σi\sigma_i: Dispersão do cluster CiC_i.
- **Índice de Calinski-Harabasz (CH)**:

  CH=tr(BK)tr(WK)⋅n−KK−1CH = \frac{\text{tr}(B_K)}{\text{tr}(W_K)} \cdot \frac{n - K}{K - 1}

  Onde:

  - tr(BK)\text{tr}(B_K): Soma das distâncias inter-clusters.
  - tr(WK)\text{tr}(W_K): Soma das distâncias intra-cluster.

---

4. Resultados Experimentais

4.1 Desempenho Geral

- **Índice de Silhueta**: Mostrou-se o mais consistente, destacando-se em cenários com clusters esféricos e baixa sobreposição.
- **Índice de Dunn**: Eficaz para clusters bem separados, mas sensível ao ruído.
- **Davies-Bouldin** e **Calinski-Harabasz**: Ótimos para clusters esféricos, mas apresentam limitações com alta dimensionalidade.

4.2 Impacto de Parâmetros

1. **Número de Clusters (KK)**:
   - Índices como Silhueta e CH são robustos para variações em KK.
2. **Ruído**:
   - Ruído significativo reduz a eficácia de quase todos os CVIs, com exceção do CH.
3. **Dimensionalidade**:
   - Alta dimensionalidade prejudica o desempenho de índices baseados em distâncias, como DB e Dunn.

---

5. Conclusões

1. **Não há um índice universalmente superior**:
   - A escolha do CVI depende da estrutura dos dados e das características dos clusters.
2. **Recomendações Práticas**:
   - Utilize o **Índice de Silhueta** para clusters esféricos e pouca sobreposição.
   - Use o **Calinski-Harabasz** em cenários com alta dimensionalidade e sem ruído extremo.
3. **Perspectivas Futuras**:
   - Estudar a adaptação dos índices para dados de alta dimensionalidade.
   - Explorar métodos híbridos que combinem os pontos fortes de diferentes índices.

---

6. Exemplo Prático

6.1 Cenário

Dados de exemplo com dois clusters esféricos:

X={(1,1),(2,2),(1,2),(10,10),(11,11),(10,11)}X = \{(1, 1), (2, 2), (1, 2), (10, 10), (11, 11), (10, 11)\}

Aplicamos o **k-means** e calculamos o Índice de Silhueta.

6.2 Implementação em Python

```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

# Dados
X = [[1, 1], [2, 2], [1, 2], [10, 10], [11, 11], [10, 11]]

# Aplicar k-means
kmeans = KMeans(n_clusters=2, random_state=0)
labels = kmeans.fit_predict(X)

# Calcular índice de silhueta
silhouette = silhouette_score(X, labels)
print(f"Índice de Silhueta: {silhouette}")
```

6.3 Implementação em C

```
#include <stdio.h>
#include <math.h>

double calculate_silhouette(double distances[][6], int labels[], int n, int k) {
    double silhouette = 0.0;
```

```
    for (int i = 0; i < n; i++) {
        double a = 0.0, b = INFINITY;
        int cluster = labels[i];

        // Calcular a(i)
        for (int j = 0; j < n; j++) {
            if (labels[j] == cluster && i != j) {
                a += distances[i][j];
            }
        }
        a /= n - 1;

        // Calcular b(i)
        for (int c = 0; c < k; c++) {
            if (c != cluster) {
                double dist_sum = 0.0, count = 0;
                for (int j = 0; j < n; j++) {
                    if (labels[j] == c) {
                        dist_sum += distances[i][j];
                        count++;
                    }
                }
                double avg_dist = dist_sum / count;
                if (avg_dist < b) b = avg_dist;
            }
        }

        silhouette += (b - a) / fmax(a, b);
    }
    return silhouette / n;
}

int main() {
    int labels[6] = {0, 0, 0, 1, 1, 1};
    double distances[6][6] = {
        {0, 1, 1.4, 12.7, 14.1, 13.4},
        {1, 0, 1.4, 12.7, 14.1, 13.4},
        {1.4, 1.4, 0, 12.7, 14.1, 13.4},
        {12.7, 12.7, 12.7, 0, 1, 1.4},
        {14.1, 14.1, 14.1, 1, 0, 1.4},
        {13.4,
```

13.4, 13.4, 1.4, 1.4, 0}, };

```
double silhouette = calculate_silhouette(distances, labels, 6, 2);
printf("Índice de Silhueta: %.2f\n", silhouette);

return 0;


}
```