Índices de Validação de Clustering - Uma Análise Abrangente e Implementação

O **agrupamento de dados** (clustering) é uma das tarefas centrais em aprendizado não supervisionado, que visa dividir um conjunto de dados em grupos homogêneos, onde os dados dentro de cada grupo ou cluster são mais semelhantes entre si do que com os dados de outros clusters. Uma das questões mais desafiadoras no contexto de clustering é **avaliar a qualidade do agrupamento**, especialmente quando os rótulos dos dados não estão disponíveis. Para resolver essa questão, utilizam-se **índices de validação de clustering** que permitem avaliar a eficácia dos resultados de agrupamento, proporcionando uma medida quantitativa da qualidade da segmentação obtida.

Este artigo explora dois índices de validação amplamente utilizados para avaliar o desempenho de algoritmos de clustering: o **Índice de Silhouette** e o **Índice de Davies-Bouldin**. Ambos fornecem as métricas que medem tanto a **coesão** quanto a **separação** entre os clusters, permitindo uma avaliação quantitativa da qualidade do agrupamento. O artigo também discute como esses índices podem ser aplicados para validar os resultados de algoritmos de clustering como **k-means**, **k-medoids** e **agregação hierárquica**.

Objetivo

O objetivo deste estudo é analisar os principais índices de validação de clustering, com foco nos índices de **Silhouette** e **Davies-Bouldin**, apresentando suas fórmulas matemáticas e discussões sobre como cada um reflete as qualidades de um bom agrupamento. O artigo também fornece uma implementação prática de ambos os índices em C++.

Índices de Validação de Clustering

1. Índice de Silhouette

O **Índice de Silhouette** é um dos índices de validação mais populares, pois mede a **coerência interna** e a **separação externa** de clusters. O índice é baseado na análise de dois parâmetros para cada ponto:

- **Coesão** (a(i)a(i)): A distância média do ponto ii para todos os outros pontos dentro do mesmo cluster.
- **Separação** (b(i)b(i)): A distância média do ponto ii para os pontos do cluster mais próximo.

O **índice de Silhouette** S(i)S(i) para um ponto ii é calculado como:

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

Onde:

- a(i)a(i) é a coesão.
- b(i)b(i) é a separação.

O valor de S(i)S(i) varia entre -1 e 1:

- **S(i)≈1S(i) \approx 1**: O ponto está bem agrupado no cluster e distante dos outros clusters.
- **S(i)≈0S(i) \approx 0**: O ponto está na fronteira entre dois clusters.
- **S(i)≈−1S(i) \approx -1**: O ponto está mal posicionado, provavelmente no cluster errado.

A média de S(i)S(i) para todos os pontos fornece uma avaliação global da qualidade do agrupamento.

2. Índice de Davies-Bouldin

O **Índice de Davies-Bouldin** (DB) mede a **separação entre os clusters** com base na relação entre a dispersão interna dos clusters e a distância entre seus centróides. Este índice é útil para avaliar o quanto os clusters são compactos internamente e bem separados externamente. O índice de Davies-Bouldin é dado por:

DB=1K∑i=1Kmaxj≠i(σi+σjd(ci,cj))DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)

Onde:

- σi\sigma_i é a dispersão (variabilidade) dentro do cluster ii.
- d(ci,cj)d(c_i, c_j) é a distância entre os centróides dos clusters ii e jj.
- KK é o número total de clusters.

Quanto **menor** o valor de DBDB, melhor a **separação entre os clusters**. Em outras palavras, um valor baixo de DBDB indica que os clusters são compactos e bem separados.

Metodologia

Para a avaliação dos índices de validação, utilizamos os seguintes algoritmos de agrupamento:

1. **k-means**: Algoritmo clássico de clustering baseado na minimização da soma das distâncias quadráticas dos pontos aos seus centróides.
2. **k-medoids**: Algoritmo de clustering semelhante ao k-means, mas com a diferença de que os centróides são pontos reais do conjunto de dados.
3. **Agrupamento Hierárquico Aglomerativo**: Algoritmo baseado na fusão iterativa dos clusters mais próximos.

Os índices de validação são então calculados e comparados para esses algoritmos em dados sintéticos e reais.

Resultados Experimentais

Dados Sintéticos

Foram gerados dados com diferentes características de separação e dispersão para avaliar o desempenho dos algoritmos de clustering e dos índices de validação. Os resultados indicaram que:

- **k-means** é mais eficiente em cenários onde a **separação entre os clusters** é clara e as variâncias das variáveis são semelhantes.
- **k-medoids** mostrou melhor desempenho quando os dados eram **menos bem comportados** ou continham **valores atípicos**, devido à sua robustez em relação a dados ruidosos.
- **Agrupamento Hierárquico** teve melhor desempenho em dados **altamente dispersos**, onde a separação entre os clusters não é esférica.

Dados Reais (UCI)

Usamos conjuntos de dados como **Iris**, **Vinho** e **Abalone** para avaliar a aplicação dos índices de Silhouette e Davies-Bouldin. O desempenho dos índices foi consistente com os resultados obtidos nos dados sintéticos, com destaque para a maior sensibilidade do índice de Silhouette em dados bem segmentados.

Código C++ - Implementação dos Índices de Validação

O código abaixo implementa os cálculos dos índices de **Silhouette** e **Davies-Bouldin** em C++, sendo um exemplo prático de como esses índices podem ser usados para avaliar o desempenho de um algoritmo de clustering.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <limits>

// Função para calcular a distância Euclidiana
double euclideanDistance(const std::vector<double>& a, const
std::vector<double>& b) {
    double sum = 0.0;
    for (size_t i = 0; i < a.size(); ++i) {
        sum += (a[i] - b[i]) * (a[i] - b[i]);
    }
    return std::sqrt(sum);
}

// Calcular a média da distância intra-cluster
double meanIntraClusterDistance(const std::vector<std::vector<double>>& data,
                                const std::vector<int>& labels, int cluster,
int pointIndex) {
    double totalDistance = 0.0;
    int count = 0;

    for (size_t i = 0; i < data.size(); ++i) {
        if (labels[i] == cluster && i != pointIndex) {
            totalDistance += euclideanDistance(data[pointIndex], data[i]);
            count++;
        }
    }

    return count > 0 ? totalDistance / count : 0.0;
}

// Calcular a distância média para o cluster mais próximo
double meanNearestClusterDistance(const std::vector<std::vector<double>>& data,
                                  const std::vector<int>& labels, int
pointIndex, int currentCluster) {
    double minDistance = std::numeric_limits<double>::max();

    for (size_t c = 0; c < data.size(); ++c) {
        if (labels[c] != currentCluster) {
            double distance = euclideanDistance(data[pointIndex], data[c]);
            if (distance < minDistance) {
                minDistance = distance;
            }
        }
    }

    return minDistance;
```

```
}

// Calcular índice de Silhouette
double silhouetteScore(const std::vector<std::vector<double>>& data, const
std::vector<int>& labels) {
    double totalScore = 0.0;
    int n = data.size();

    for (int i = 0; i < n; ++i) {
        int currentCluster = labels[i];

        // Coesão (a)
        double a = meanIntraClusterDistance(data, labels, currentCluster, i);

        // Separação (b)
        double b = meanNearestClusterDistance(data, labels, i, currentCluster);

        // Silhouette
        double s = (b - a) / std::max(a, b);
        totalScore += s;
    }

    return totalScore / n;
}

// Calcular índice de Davies-Bouldin
double daviesBouldinScore(const std::vector<std::vector<double>>& data, const
std::vector<int>& labels, int k) {
    std::vector<double> clusterDispersion(k, 0.0);
    std::vector<std::vector<double>> centroids(k,
std::vector<double>(data[0].size(), 0.0));
    std::vector<int> clusterSizes(k, 0);

    // Calcular centróides e dispersões
    for (size_t i = 0; i < data.size(); ++i) {
        int cluster =
```

labels[i]; clusterSizes[cluster]++; for (size_t j = 0; j < data[0].size(); ++j) { centroids[cluster][j] += data[i][j]; } }

```
for (int c = 0; c < k; ++c) {
    for (size_t j = 0; j < data[0].size(); ++j) {
        centroids[c][j] /= clusterSizes[c];
    }
}

for (size_t i = 0; i < data.size(); ++i) {
    int cluster = labels[i];
    clusterDispersion[cluster] += euclideanDistance(data[i],
centroids[cluster]);
}

for (int c = 0; c < k; ++c) {
    clusterDispersion[c] /= clusterSizes[c];
}

// Calcular índice Davies-Bouldin
double totalScore = 0.0;

for (int i = 0; i < k; ++i) {
    double maxRatio = 0.0;
    for (int j = 0; j < k; ++j) {
        if (i != j) {
```

```
            double distance = euclideanDistance(centroids[i], centroids[j]);
            double ratio = (clusterDispersion[i] + clusterDispersion[j]) /
distance;
            maxRatio = std::max(maxRatio, ratio);
        }
    }
    totalScore += maxRatio;
}

return totalScore / k;


}
```

int main() { // Dados simulados std::vector<std::vector> data = { {1.0, 2.0}, {1.5, 1.8}, {5.0, 8.0}, {8.0, 8.0}, {1.0, 0.6}, {9.0, 11.0}, {8.0, 2.0}, {10.0, 2.0} };

```
// Rótulos de cluster gerados por um algoritmo de clustering (exemplo)
std::vector<int> labels = {0, 0, 1, 1, 0, 1, 2, 2};

int k = 3; // Número de clusters

// Calcular índices de validação
double silhouette = silhouetteScore(data, labels);
double daviesBouldin = daviesBouldinScore(data, labels, k);

std::cout << "Índice de Silhouette: " << silhouette << "\n";
std::cout << "Índice de Davies-Bouldin: " << daviesBouldin << "\n";

return 0;


}
```

Explicação Completa do Código:

O código acima calcula os dois índices de validação: Silhouette** e Davies-Bouldin, usados para avaliar a qualidade de agrupamentos gerados por algoritmos como k-means.

Funções Principais:

1. `euclideanDistance`: Calcula a distância Euclidiana entre dois pontos \(a\) e \(b\). A fórmula usada é a soma das diferenças quadradas das coordenadas dos pontos, seguida pela raiz quadrada da soma.

2. `meanIntraClusterDistance`: Calcula a média das distâncias entre um ponto \(i\) e todos os outros pontos do mesmo cluster. Isso é usado para calcular a **coesão** no Índice de Silhouette.

3. `meanNearestClusterDistance`: Calcula a distância média entre um ponto \(i\) e todos os pontos do cluster mais próximo. Isso é usado para calcular a **separação** no Índice de Silhouette.

4. `silhouetteScore`: Calcula o Índice de Silhouette, que mede a qualidade do agrupamento com base na coesão e separação dos pontos em relação aos seus clusters.

5. `daviesBouldinScore`: Calcula o Índice de Davies-Bouldin, que mede a separação entre os clusters com base na dispersão interna de cada cluster e na distância entre seus centróides.

6. `main`: Inicializa os dados simulados e os rótulos de cluster e então calcula e exibe os índices de validação.

Resultados:

Índice de Silhouette: 0.665126
Índice de Davies-Bouldin: 0.382797