Artigo 5: "Cluster Ensembles: A Survey of Approaches with Recent Extensions and Applications" (Boongoen & Iam-On, 2018)

---

1. Introdução

Os **ensembles de clusters** têm emergido como uma técnica poderosa para melhorar a robustez e a qualidade dos resultados de agrupamento. Em vez de confiar em um único algoritmo de clustering, os ensembles combinam múltiplas partições geradas por diferentes algoritmos ou configurações para produzir uma **partição consensual**. Essa abordagem é amplamente utilizada em diversas aplicações, como biologia computacional, mineração de dados e sistemas de recomendação.

Este artigo oferece uma revisão abrangente sobre os ensembles de clustering, abordando:

1. As principais estratégias para gerar partições.
2. Métodos de consenso para combinar partições.
3. Extensões recentes e aplicações práticas.

---

2. Objetivo do Estudo

O objetivo principal é fornecer uma visão detalhada das metodologias de ensembles de clustering, explorando:

- A eficácia das abordagens existentes.
- As limitações enfrentadas na prática.
- Extensões que lidam com novos desafios, como dados em larga escala, múltiplas visões e clusters sobrepostos.

---

3. Componentes dos Ensembles de Clustering

Os ensembles de clustering consistem em duas etapas principais: **geração de partições** e **funções de consenso**.

3.1 Geração de Partições

A diversidade das partições é essencial para o sucesso dos ensembles. As estratégias incluem:

1. **Homogêneas**:
    - Múltiplas execuções do mesmo algoritmo (ex.: k-means) com inicializações ou parâmetros diferentes.
    - Benefício: Simplicidade e alta compatibilidade.
    - Limitação: Baixa diversidade nas partições.
2. **Heterogêneas**:
    - Combinação de diferentes algoritmos de clustering (ex.: k-means, DBSCAN, hierárquico).
    - Benefício: Alta diversidade.
    - Limitação: Complexidade computacional.
3. **Baseadas em Subespaços ou Amostragem**:

- Aplicação de algoritmos em subconjuntos de dados ou atributos.
- Benefício: Reduz a dimensionalidade e aumenta a diversidade.
- Limitação: Pode negligenciar informações globais.

3.2 Funções de Consenso

As funções de consenso combinam as partições para produzir a solução final. Principais abordagens incluem:

1. **Baseadas em Similaridade Pareada**:
    - Constroem uma **matriz de co-ocorrência** SS, onde S[i][j]S[i][j] representa a frequência de xix_i e xjx_j estarem no mesmo cluster em diferentes partições.
    - Exemplo:
        - Partição 1: π1={{x1,x2},{x3,x4}}\pi_1 = \{\{x_1, x_2\}, \{x_3, x_4\}\}
        - Partição 2: π2={{x1,x3},{x2,x4}}\pi_2 = \{\{x_1, x_3\}, \{x_2, x_4\}\}
        - Matriz de Similaridade: S=[10.50.500.5100.50.5010.500.50.51]S = \begin{bmatrix} 1 & 0.5 & 0.5 & 0 \\ 0.5 & 1 & 0 & 0.5 \\ 0.5 & 0 & 1 & 0.5 \\ 0 & 0.5 & 0.5 & 1 \end{bmatrix}
2. **Baseadas em Grafos**:
    - Representam partições como grafos ponderados, onde os vértices são objetos e as arestas representam similaridade.
    - Métodos como corte mínimo identificam clusters.
3. **Baseadas em Reagrupamento**:
    - Reagrupam os clusters de todas as partições em uma única solução usando algoritmos como k-means.

---

4. Desafios e Extensões

4.1 Desafios

- **Diversidade e Qualidade**:
    - Partições pouco diversificadas reduzem o benefício do ensemble.
- **Complexidade Computacional**:
    - A combinação de partições grandes exige métodos eficientes.
- **Clusters Sobrepostos e Ruidosos**:
    - Métodos tradicionais têm dificuldade em lidar com essas características.

4.2 Extensões Recentes

1. **Dados de Múltiplas Visões**:
    - Aplicações em cenários onde diferentes fontes de dados descrevem os mesmos objetos (ex.: imagem e texto).
    - Estratégia: Combinar partições geradas em cada visão separadamente.
2. **Clusters Dinâmicos**:
    - Métodos que adaptam os ensembles em fluxos de dados dinâmicos.

3. **Aplicações em Grande Escala**:
    - Uso de técnicas distribuídas e paralelas para lidar com conjuntos de dados massivos.

---

5. Aplicações Práticas

1. **Biologia Computacional**:
    - Análise de expressão gênica em bioinformática.
    - Descoberta de padrões em redes metabólicas.
2. **Sistemas de Recomendação**:
    - Personalização em e-commerce com base em perfis de usuários.
3. **Análise de Imagens e Texto**:
    - Segmentação de imagens médicas.
    - Agrupamento de documentos em grandes corpora.

---

6. Exemplo Prático

6.1 Cenário

Considere um conjunto de dados $X=\{x1,x2,x3,x4\}X = \{x_1, x_2, x_3, x_4\}$ e duas partições:

- $\pi1=\{\{x1,x2\},\{x3,x4\}\}\pi_1 = \{\{x_1, x_2\}, \{x_3, x_4\}\}$
- $\pi2=\{\{x1,x3\},\{x2,x4\}\}\pi_2 = \{\{x_1, x_3\}, \{x_2, x_4\}\}$

objetivo é combinar as partições usando uma **matriz de similaridade pareada** e gerar a partição consensual.

6.2 Implementação em Python

```
import numpy as np
from sklearn.cluster import KMeans

# Dados e partições
n = 4
partitions = [
    [[0, 1], [2, 3]],
    [[0, 2], [1, 3]]
]

# Construir matriz de similaridade
S = np.zeros((n, n))
for partition in partitions:
    for cluster in partition:
        for i in cluster:
            for j in cluster:
                S[i, j] += 1
S /= len(partitions)

# Aplicar k-means na matriz de similaridade
kmeans = KMeans(n_clusters=2, random_state=0)
labels = kmeans.fit_predict(S)
print("Clusters finais:", labels)
```

6.3 Implementação em C

```
#include <stdio.h>
#define N 4

void update_similarity(double S[N][N], int partitions[][2], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < 2; j++) {
            for (int k = 0; k < 2; k++) {
                S[partitions[i][j]][partitions[i][k]] += 1.0;
            }
        }
    }
}

int main() {
    double S[N][N] = {0};
    int partitions[2][2] = {{0, 1}, {2, 3}, {0, 2}, {1, 3}};

    // Atualizar matriz de similaridade
    update_similarity(S, partitions, 2);

    // Normalizar matriz
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            S[i][j] /= 2.0;
        }
    }

    // Exibir matriz de similaridade
    printf("Matriz de Similaridade:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%.2f ", S[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

---

7. Conclusão

1. **Benefícios dos Ensembles**:
    - Melhoram a robustez e a generalização das soluções de clustering.
    - Lidam bem com variabilidade entre algoritmos ou configurações.
2. **Limitações**:
    - Necessitam de diversidade nas partições para serem eficazes.
    - Métodos eficientes são críticos em cenários com grandes volumes de dados.
3. **Perspectivas Futuras**:
    - Explorar combinações híbridas (ex.: baseadas em similaridade pareada e grafos).
    - Aplicações em clustering dinâmico e de múltiplas visões.