K-Means em Julia

February 11, 2016

1 Trabalho de Implementação

1.1 INF2912 - Otimização Combinatória

1.1.1 Prof. Marcus Vinicius Soledade Poggi de Aragão

1.1.2 2015-2

1.1.3 Ciro Cavani

BigData / Globo.com Algoritmos de clusterização.

1.2 Conteúdo

Esse notebook tem o desenvolvimento e avaliação do algoritmo iterativo do K-Means (algoritmo de Lloyd).

A avaliação do algoritmo é baseada em um mapeamento entre a maioria dos itens que foram atribuídos a um determinado cluster e o correspondente os valores verdadeiros gerados nesse cluster.

O K-Means teve resultados muito bons.

1.3 Dataset

```
In [1]: include("../src/clustering.jl")
        import Inf2912Clustering
        const Clustering = Inf2912Clustering
        dataset = Clustering.load_dataset("small")
        Clustering.summary(dataset)
        sleep(0.2)
```

```
Number of Groups: 3
Number of Features: 200
Number of Features (group): 40
Probability of Activation: 0.8
Number of Objects (total): 100
Number of Objects per Group (min): 7
Number of Objects per Group (max): 66
Number of Objects in 1: 15
Number of Objects in 2: 39
Number of Objects in 3: 46
```

1.4 K-Means

Consiste em executar o algoritmo K-means determinar os pontos centrais de cada grupo e classificar cada objeto como sendo do grupo com ponto central mais próximo

https://en.wikipedia.org/wiki/K-means_clustering

1.4.1 Algoritmo Iterativo

1. Choose k cluster centers randomly generated in a domain containing all the points,
2. Assign each point to the closest cluster center,
3. Recompute the cluster centers using the current cluster memberships,
4. If a convergence criterion is met, stop; Otherwise go to step 2.

In [2]:
```
function kmeans(dataset, k; maxiters=20)
    inputs = map(v -> float(v[1]), dataset.data)

    # inicialização com amostragem sem reposição de k objetos como centros iniciais
    means = map(i -> inputs[i], randperm(length(inputs))[1:k])

    "função que calcula o índice do centro de menor distância de v"
    classify(v) = indmin(map(c -> norm(c - v), means))

    assignments::Array{Int,1} = []
    iters = 0

    while iters < maxiters
        iters += 1

        # calcula o centro associado a cada objeto
        new_assignments = map(classify, inputs)

        # encerra o processamento se não tiver mudança com a última iteração
        assignments == new_assignments && break

        # recalcula os centros como a média dos pontos do último agrupamento
        assignments = new_assignments

        #println("Centros ", iters, ": ", means)
        #println("Agrupamentos ", iters, ": ", new_assignments)

        for i=1:k
            # lista todos os objetos do i-ésimo agrupamento
            i_points = map(ii -> inputs[ii], findin(assignments, i))

            isempty(i_points) && continue
            means[i] = mean(i_points)
        end
    end

    assignments
end

kmeans(dataset, 3)
```

Out[2]:
```
100-element Array{Int64,1}:
 1
 2
 1
 1
 3
 3
```

```
 1
 1
 1
 1
 3
 1
 1
 ⋮
 1
 3
 2
 3
 1
 2
 3
 3
 1
 3
 3
 3
```

```
In [3]: function kmeans_approx(dataset, k)
            assignments = kmeans(dataset, k)
            centermap = Clustering.mapping(dataset, assignments, k)
            map(c -> centermap[c], assignments)
        end

        let
            n = 100
            k = 3
            c = 16
            c_y = 3

            tiny = Clustering.Dataset(size=n, groups=k, features=c, slot=c_y)

            prediction = kmeans_approx(tiny, k)
            Clustering.evaluation_summary(tiny, prediction)
        end
```

```
Precision: 95.79%
Recall: 94.79%
F-score: 0.95

Número de predições: 100
Acertos: 91 (91.0%)
Falso negativo: 5 (5.0%)
Falso positivo: 4 (4.0%)

Cluster 1

Objetos: 36
Accuracy: 95.0%
Precision: 96.97%
Recall: 88.89%
```

```
F-score: 0.93

Acerto positivo: 32 (88.89%)
Acerto negativo: 63 (98.44%)
Falso negativo: 4 (80.0%)
Falso positivo: 1 (25.0%)

Cluster 2

Objetos: 23
Accuracy: 92.0%
Precision: 77.78%
Recall: 91.3%
F-score: 0.84

Acerto positivo: 21 (91.3%)
Acerto negativo: 71 (92.21%)
Falso negativo: 2 (40.0%)
Falso positivo: 6 (150.0%)

Cluster 3

Objetos: 41
Accuracy: 95.0%
Precision: 95.0%
Recall: 92.68%
F-score: 0.94

Acerto positivo: 38 (92.68%)
Acerto negativo: 57 (96.61%)
Falso negativo: 3 (60.0%)
Falso positivo: 2 (50.0%)
```