K-Means em Julia

February 14, 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
```

```
WARNING: redefining constant srcdir
WARNING: redefining constant default_datasetdir
```

```
Out[1]: Inf2912Clustering
```

```
In [2]: dataset = Clustering.dataset_tiny()
        Clustering.summary(dataset)
        sleep(0.2)
```

```
Clusters: 3
Dimension (features): 16
Features per Cluster: 3
Probability of Activation: 0.8

Size: 100
Min Cluster size: 20
Max Cluster size: 40
Cluster 1 size: 23
Cluster 2 size: 41
Cluster 3 size: 36
```

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 [3]: import Clustering: Input, Dataset

        "Algoritmo de clusterização K-Means (algoritmo de Lloyd)."
        function kmeans(input::Input, k::Int; maxiters=20)
            inputs = map(v -> float(v), input.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::Dataset, k::Int) = kmeans(dataset.input, k)

         kmeans(dataset, 3)
```

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

```
In [4]: import Clustering.mapping

        "Algoritmo de clusterização K-Means (algoritmo de Lloyd) \
        aproximado para os grupos pré-definidos do dataset."
        function kmeans_approx(dataset::Dataset, k::Int)
            assignments = kmeans(dataset, k)
            centermap = mapping(dataset, assignments, k)
            map(c -> centermap[c], assignments)
        end

        let
            k = dataset.clusters
            @time prediction = kmeans_approx(dataset, k)
            Clustering.evaluation_summary(dataset, prediction; verbose=true)
            sleep(0.2)
        end
```

```
0.109878 seconds (123.90 k allocations: 6.151 MB)
Confusion Matrix:

[22 0 1
 0 36 5
 1 3 32]

Size: 100
Correct: 90
Mistakes: 10
Accuracy: 90.0%

Cluster 1

Size: 23
Accuracy: 98.0%
Precision: 95.65%
Recall: 95.65%
F-score: 0.96

True Positive: 22 (95.65%)
True Negative: 76 (98.7%)
False Negative: 1 (10.0%)
False Positive: 1 (10.0%)

Cluster 2

Size: 41
Accuracy: 92.0%
Precision: 92.31%
Recall: 87.8%
F-score: 0.9

True Positive: 36 (87.8%)
True Negative: 56 (94.92%)
False Negative: 5 (50.0%)
False Positive: 3 (30.0%)

Cluster 3

Size: 36
Accuracy: 90.0%
Precision: 84.21%
Recall: 88.89%
F-score: 0.86

True Positive: 32 (88.89%)
True Negative: 58 (90.62%)
False Negative: 4 (40.0%)
False Positive: 6 (60.0%)
```

```
In [5]: Clustering.test_dataset("small", kmeans_approx)
        sleep(0.2)
```

```
0.020264 seconds (43.69 k allocations: 21.656 MB, 32.31% gc time)
Confusion Matrix:

[367 0 0
 0 266 0
 0 0 367]

Size: 1000
Correct: 1000
Mistakes: 0
Accuracy: 100.0%

Cluster 1

Size: 367
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 367 (100.0%)
True Negative: 633 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)

Cluster 2

Size: 266
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 266 (100.0%)
True Negative: 734 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)

Cluster 3

Size: 367
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 367 (100.0%)
True Negative: 633 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)
```

```
In [6]: Clustering.test_dataset("large", kmeans_approx)
        sleep(0.2)
```

```
0.518594 seconds (448.74 k allocations: 216.570 MB, 17.72% gc time)
Confusion Matrix:

[3814 0 0
 0 3973 0
 0 0 2213]

Size: 10000
Correct: 10000
Mistakes: 0
Accuracy: 100.0%

Cluster 1

Size: 3814
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 3814 (100.0%)
True Negative: 6186 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)

Cluster 2

Size: 3973
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 3973 (100.0%)
True Negative: 6027 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)

Cluster 3

Size: 2213
Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-score: 1.0

True Positive: 2213 (100.0%)
True Negative: 7787 (100.0%)
False Negative: 0 (NaN%)
False Positive: 0 (NaN%)
```