K-Means em Julia

February 12, 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)
```

```
WARNING: type Dataset not present in workspace; reconstructing


        LoadError: MethodError: ‘summary‘ has no method matching summary(::JLD.##Dataset#8091)
    you may have intended to import Base.summary
    while loading In[1], in expression starting on line 5
```

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}:
 2
 2
 3
 3
 1
 3
```

```
 3
 2
 3
 3
 3
 3
 3
 ⋮
 1
 3
 3
 3
 3
 3
 3
 3
 1
 1
 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; verbose=true)
        end
```

```
Matriz de Confusão:

[9 7 0
 2 14 0
 17 8 43]

Tamanho: 100
Acertos: 66
Erros: 34
Accuracy: 66.0%

Cluster 1

Tamanho: 16
Accuracy: 74.0%
```

```
Precision: 32.14%
Recall: 56.25%
F-score: 0.41

Acerto positivo: 9 (56.25%)
Acerto negativo: 65 (77.38%)
Falso negativo: 7 (20.59%)
Falso positivo: 19 (55.88%)

Cluster 2

Tamanho: 16
Accuracy: 83.0%
Precision: 48.28%
Recall: 87.5%
F-score: 0.62

Acerto positivo: 14 (87.5%)
Acerto negativo: 69 (82.14%)
Falso negativo: 2 (5.88%)
Falso positivo: 15 (44.12%)

Cluster 3

Tamanho: 68
Accuracy: 75.0%
Precision: 100.0%
Recall: 63.24%
F-score: 0.77

Acerto positivo: 43 (63.24%)
Acerto negativo: 32 (100.0%)
Falso negativo: 25 (73.53%)
Falso positivo: 0 (0.0%)
```