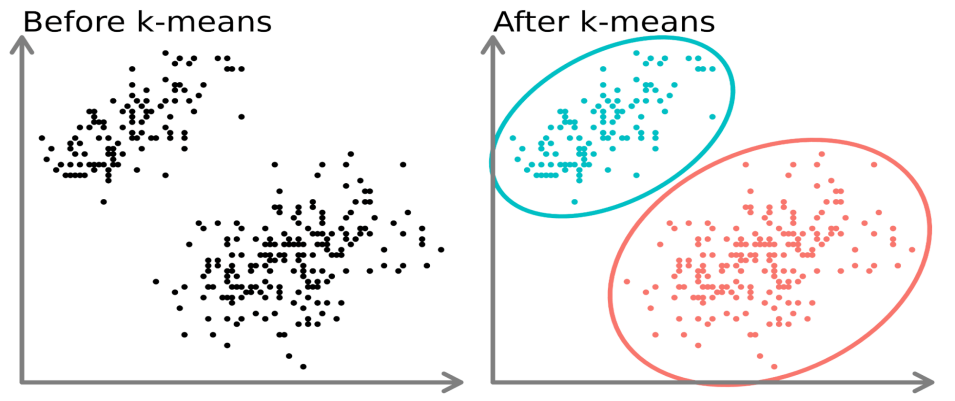
Agrupamento k-means em RISC-V

Este documento descreve o projeto da cadeira de Introdução à Arquitetura de Computadores 2023/2024 (LEIC-A e LEIC-T).

1 Introdução

O objetivo do projeto é desenvolver, em *assembly* RISC-V, um programa que, dado um conjunto inicial de pontos num espaço bidimensional (2D), identifica k grupos de pontos, tendo em conta a proximidade relativa dos pontos de cada grupo. Para tal, será implementado o algoritmo iterativo *k-means*.



Créditos da imagem: https://www.datacamp.com

Este algoritmo é amplamente usado em aplicações de diferentes domínios – desde visão computacional, aprendizagem automática, deteção de intrusões em cibersegurança, a astronomia. Para que essas aplicações sejam eficazes, é essencial que a implementação do algoritmo *k-means* seja eficiente.

Uma boa descrição do algoritmo pode ser encontrada aqui:

https://statquest.org/statquest-k-means-clustering/

O programa final a desenvolver neste projeto recebe os seguintes *inputs* principais:

- *points*, um vetor de pontos num espaço 2D (cada ponto consiste num par de cordenadas, {x,y});
- n, o número de pontos, ou seja a dimensão do vetor `points`;
- k, o número de agrupamentos (*clusters*) a considerar, ou seja o parâmetro k do algoritmo *k-means*.
- l, o número máximo de iterações do algoritmo.

A partir destes elementos, o algoritmo $k-means$ é executado uma única vez. Ou seja, fica fora do projeto a otimização de executar múltiplas instâncias do algoritmo e a escolha daquela que minimiza a variância.

Durante a execução, os k agrupamentos e os respetivos centróides (*centroids*) devem ser apresentados num ecrã, como pontos coloridos (com k cores distintas) numa matriz 2D. À medida que cada iteração é calculada e ajusta os agrupamentos e os respetivos centróides, essa informação vai sendo atualizada no ecrã.

2 Organização do projeto

O projeto é dimensionado para ser feito em grupos de 3 estudantes. O projeto está decomposto num conjunto de problemas, sendo cada problema resolvido por um procedimento (ou função).

Recomendamos que cada grupo implemente e teste cada problema/procedimento antes de passar ao próximo. Isso permitirá um progresso gradual até à solução.

Antes da entrega final, existe uma entrega preliminar. Na entrega preliminar, é esperado que cada grupo já tenha uma solução que é capaz de apresentar o conjunto inicial de pontos no ecrã e também o ponto médio (i.e., o centróide) relativo a esse conjunto (total) de pontos. No fundo, isto corresponde ao resultado que se obteria correndo o algoritmo *k-means* com $k = 1$.

Nas secções seguintes, apresentamos os procedimentos que compõem a solução final, pela ordem que sugerimos para serem implementados. Na vossa solução têm de respeitar esta organização em procedimentos.

Este enunciado é complementado por um esqueleto inicial do código do projeto, que será disponibilizado no site da disciplina antes do arranque do projeto. Esse código inicial define define uma estrutura de código e de organização de dados em memória que cada grupo terá de respeitar quando resolver o projeto.

3 Estado e procedimentos

Além dos *inputs* definidos anteriormente, o programa deve manter o seguinte estado:

- $clusters[n]$, um vetor que indica a qual agrupamento (0 a $k-1$) a que cada ponto $points[i]$ está associado.
- $centroids[k]$, um vetor que indica as coordenadas dos centróides dos k agrupamentos. Tal como o vetor $points$, cada elemento do vetor $centroids$ tem um par de coordenadas 2D.
- entre outros (a definir pelo grupo).

De seguida começamos por descrever os **procedimentos que são necessários para a entrega preliminar**.

`cleanScreen`

Limpa todos os pontos do ecrã.

`printClusters`

Caso $k = 1$, este procedimento percorre o array $points$ e pinta o ponto correspondente no ecrã. **Nota: na entrega preliminar, só este caso precisa estar implementado.**

Caso $k > 1$, este procedimento consulta o vetor $clusters$ para determinar qual o agrupamento a que cada um dos n pontos está associado. Cada ponto é pintado no ecrã com uma cor associada a esse agrupamento. A tabela que associa cores a cada agrupamento é fornecida no código inicial do projeto.

`printCentroids`

Este procedimento percorre o vetor $centroids$ e, para cada centróide, pinta-o no ecrã. A tabela que define as cores de cada centróide é fornecida no código inicial do projeto.

`calculateCentroids`

Caso $k = 1$, este procedimento percorre o vetor $points$ e, para cada dimensão, calcula a média das coordenadas nessa dimensão de todos os pontos. Este par de inteiros (a média da dimensão x e a média da dimensão y) determina as coordenadas do centróide do conjunto total de pontos. Preenche ambas as médias na primeira entrada do vetor $centroids$. **Nota: na entrega preliminar, só este caso precisa estar implementado.**

Caso $k > 1$, este procedimento usa o vetor $clusters$ para determinar quais pontos atualmente estão associados a cada agrupamento (entre os k agrupamentos). Para cada agrupamento, calcula o centróide dos pontos desse agrupamento (do mesmo modo que descrito acima, para $k = 1$). Preenche as coordenadas de cada um dos centróides no vetor $centroids$.

`mainSingleCluster`

Este é o procedimento principal do projeto a submeter na entrega preliminar.

Deve executar a seguinte sequência de passos:

1. Coloca 'k=1'.
2. `cleanScreen`.
3. `printClusters`.
4. `calculateCentroids`.
5. `printCentroids`.
6. Termina.

Os procedimentos seguintes só são necessários para a entrega final.

`initializeCentroids`

Este procedimento inicializa os valores iniciais do vetor *centroids*. Cada um dos k centróides deve ser colocado num par de coordenadas escolhido de forma pseudo-aleatória.

`manhattanDistance`

Dados dois pontos (passados como argumento), este procedimento retorna a distância de Manhattan entre ambos.

`nearestCLuster`

Dado um ponto (passado como argumento), consulta o vetor *centroids* e escolhe aquele que está à menor distância (de Manhattan) deste ponto. Retorna o índice (de 0 a $k-1$) do agrupamento associado a esse centróide.

`mainKMeans`

Executa o algoritmo *k-means* (apenas uma instância do algoritmo). No início de cada iteração, o ecrã deve ser limpo. No final da iteração, o ecrã deve ser preenchido com o estado atual após essa iteração (pontos coloridos de acordo com o seu agrupamento, além dos centróides de cada agrupamento).

O algoritmo deve terminar assim que uma iteração não altere a posição de nenhum centróide, ou assim que l iterações tenham sido executadas.

4 Considerações práticas

O projeto deve ser desenvolvido no simulador Ripes, que será apresentado nas aulas laboratoriais.

O processador deve ser um RISC-V de 32 bits, ou seja aquele que será estudado nas aulas teóricas e laboratoriais da disciplina.

O ecrã deve ser um dispositivo `LED Matrix`, que pode ser criado na secção *I/O* do simulador Ripes.

Todos os cálculos de coordenadas e distâncias devem recorrer exclusivamente a valores inteiros. Ou seja, não devem ser usados valores (nem instruções) de vírgula flutuante. Essa matéria está fora do âmbito da disciplina.

5 Entregas e avaliação

O projeto é submetido pelo fénix.

- Entrega preliminar (10% da nota final a IAC): 17/5/2024, 23h59.
- Entrega final (30% da nota final a IAC): 31/5/2024, 23h59.

Nota mínima de 8,0.

Usaremos software de deteção de cópias. Casos de plágio seguem procedimentos do regulamento de avaliação do IST.