Felipe Fill Cardoso 113.305–7

K Nearest Neighbours

O objetivo deste trabalho é demonstrar o funcionamento do algoritmo K Nearest Neighbours (KNN) aplicado a um cenário específico: calcular a provável nota de um aluno que utilizou n livros e compareceu a x aulas de um curso, baseando-se em um conjunto de treinamento (training set) fornecido.

Algoritmo

Dada uma entrada x (qualquer tipo, desde que de acordo com o conjunto de treinamento) e um inteiro k, o algoritmo deve buscar os k elementos com o valor de x mais próximos da entrada; com os valores selecionados, é feita a somatória dos mesmos e esse valor é dividido por k (média aritmética). Esse novo valor é a saída do algoritmo.

Implementação

O conjunto de treinamento fornecido para o desenvolvimento do trabalho contém valores distribuídos em três colunas: books (livros), attends (presenças) e grade (nota). Os dois primeiros são a entrada.

Para calcular a proximidade dos dados de entrada do algoritmo com os do conjunto de treinamento, foi utilizada a formula da distância euclidiana.

Estrutura do projeto

O projeto é dividido em três classe distintas:

- `Dataset`: classe que representa o conjunto de treinamentos.
- `DatasetReader`: classe estática que faz a leitura do conjunto de um arquivo para a classe `Dataset`.
- `KNearestNeighbours`: classe que representa o algoritmo KNN.

A relação entre as classes pode ser vista no diagrama de classes.

Descrição das classes

Dataset

- `Dataset()`: construtor padrão da classe.
- `void AddLine(int books, int attends, int grade)`: adiciona uma linha ao conjunto de dados.

DatasetReader

- `static Dataset read(const char * path)`: lê conjunto de dados a partir do arquivo disponível no caminho `path`.

KNearestNeighbours

- `KNearestNeighbours(const char * training_set_path)`: construtor que recebe o caminho do conjunto de treinamento.
- `vector<int> SortForCombination(int books, int attends)`: gera um vetor com os resultados ordenados por proximidade da entrada, método privado.
- `float Estimate(int books, int attends, int k)`: estima o valor para a entrada com a quantidade de elementos `k`.

Resultados

O projeto tinha o objetivo de resolver alguns exemplos utilizando diferentes entradas e números de exemplos (K). Segue resultados:

```
 Printing for k = 1
Books: 0     Attends:     5    Grade Estimation: 61
Books: 4     Attends:    20    Grade Estimation: 88
Books: 2     Attends:    10    Grade Estimation: 47
Books: 4     Attends:    15    Grade Estimation: 89


 Printing for k = 2
Books: 0     Attends:     5    Grade Estimation: 63.5
Books: 4     Attends:    20    Grade Estimation: 87.5
Books: 2     Attends:    10    Grade Estimation: 44
Books: 4     Attends:    15    Grade Estimation: 72
```

```
 Printing for k = 3
Books: 0     Attends:     5    Grade Estimation: 61
Books: 4     Attends:    20    Grade Estimation: 86
Books: 2     Attends:    10    Grade Estimation: 43.6667
Books: 4     Attends:    15    Grade Estimation: 71


 Printing for k = 5
Books: 0     Attends:     5    Grade Estimation: 58.8
Books: 4     Attends:    20    Grade Estimation: 88.8
Books: 2     Attends:    10    Grade Estimation: 47
Books: 4     Attends:    15    Grade Estimation: 73.2


 Printing for k = 10
Books: 0     Attends:     5    Grade Estimation: 54.9
Books: 4     Attends:    20    Grade Estimation: 72.9
Books: 2     Attends:    10    Grade Estimation: 51.8
Books: 4     Attends:    15    Grade Estimation: 72.5
```