ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Cấu trúc dữ liệu và giải thuật - CO2003

Bài tập lớn 1

XỬ LÍ DỮ LIỆU MNIST VÀ HIỆN THỰC GIẢI THUẬT kNN BẰNG CẤU TRÚC DỮ LIỆU DANH SÁCH

TP. HỒ CHÍ MINH, THÁNG 02/2024

ĐẶC TẢ BÀI TẬP LỚN

Phiên bản 1.0

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên ôn lại và sử dụng thành thục:

- Lập trình hướng đối tượng.
- Các cấu trúc dữ liệu danh sách.
- Giải thuật sắp xếp.

2 Dẫn nhập

Bài toán phân loại là một trong những bài toán cơ bản và phổ biến nhất trong học máy. Mục tiêu của bài toán là phân chia dữ liệu thành các nhóm (lớp) khác nhau dựa trên các đặc điểm của dữ liệu. Một số bài toán đơn giản của bài toán phân loại có thể kể đến như: phân loại email là thư rác hay thư chính, phân loại hình ảnh là ảnh mèo hay ảnh chó, v.v... Trong bài tập lớn này, sinh viên được làm quen với giải thuật phân loại đơn giản k-nearest neighbors (kNN), và ứng dụng giải thuật này để xử lí và phân loại tập dữ liệu MNIST.

Sinh viên được yêu cầu hiện thực một class cung cấp cho người dùng khả năng xử lý tập dữ liệu MNIST, đồng thời một class hiện thực thuật toán phân loại kNN một cách hiệu quả sử dụng các cấu trúc dữ liệu danh sách.

3 Mô tả

3.1 Tập dữ liệu MNIST

Tập dữ liệu MNIST là một trong những tập dữ liệu phổ biến nhất và thường được sử dụng để thử nghiệm và đánh giá các thuật toán học máy và thị giác máy tính. Tên gọi "MNIST" là viết tắt của "Modified National Institute of Standards and Technology". Tập dữ liệu này chứa một bộ các hình ảnh được chụp từ các mẫu viết tay của các chữ số từ 0 đến 9, mỗi hình ảnh có kích thước 28x28 pixel. Một số ví dụ về các ảnh trong tập dữ liệu được mô tả ở Hình 1

Dưới đây là một số đặc điểm quan trọng của tập dữ liệu MNIST:

Hình 1: Ví dụ về các mẫu dữ liệu trong tập dữ liệu MNIST

- Số lượng mẫu: Tập dữ liệu MNIST bao gồm 60,000 hình ảnh trong tập huấn luyện và 10,000 hình ảnh trong tập kiểm tra. Tuy nhiên, để giảm bớt khối lượng tính toán, số lượng dữ liệu dùng để huấn luyện và kiểm tra sẽ được thu nhỏ lại. Số lượng hình ảnh tối đa được sử dụng sẽ được thông báo sau.
- Loại dữ liệu: Mỗi hình ảnh trong tập dữ liệu MNIST được mã hóa dưới dạng các giá trị pixel, trong đó giá trị của mỗi pixel nằm trong phạm vi từ 0 đến 255, thể hiện độ sáng của pixel.
- Phân loại: Mỗi hình ảnh đại diện cho một trong các chữ số từ 0 đến 9. Mục tiêu là phải xây dựng một mô hình có khả năng phân loại chính xác chữ số trong hình ảnh.

Sinh viên được cung cấp file ***mnist.csv***, trong đó:

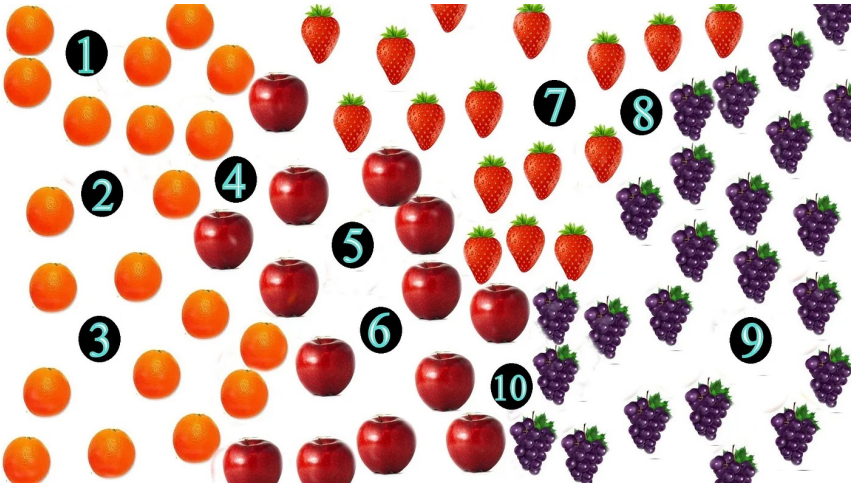
- Mỗi dòng là mô tả một hình ảnh chữ số, được mã hóa thành 28x28 = 714 cột và một cột nhãn.
- Cột nhãn (label) đánh nhãn cho số được thể hiện bởi hình ảnh. Đối với mỗi ảnh, khoảng giá trị của ô trong cột này là từ 0 đến 9.
- Các cột còn lại, có tiêu đề ixj trong đó i, j nằm trong khoảng [1,28] thể hiện tọa độ của điểm ảnh. Đối với mỗi ảnh, khoảng giá trị của ô trong cột này là từ 0 đến 255 (độ đậm của điểm ảnh).

3.2 Thuật toán k-nearest neighbors

Thuật toán kNN là một thuật toán phân loại đơn giản nhưng hiệu quả, dựa trên việc tìm kiếm k điểm dữ liệu gần nhất với điểm dữ liệu mới cần phân loại.

3.2.1 Ý tưởng

Để dễ nắm bắt ý tưởng của thuật toán, ta cùng giải một câu đố nhỏ ở Hình 2. Cửa hàng của Bob có 4 loại trái cây: Táo, Cam, Nho và Dâu. Biết rằng cửa hàng Bob bày biện các trái cây như hình sau:



Hình 2: Các trái cây trong cửa hàng của Bob

Nhiệm vụ của bạn là phải dự đoán các trái cây được đánh dấu từ 1 đến 10 sẽ là trái cây gì. Ta có được các suy luận bằng mắt thường như sau:

- 1,2,3 → Cam
- 4 → Không chắc liệu đó là Cam hay Táo
- 5,6 → Táo
- 7 → Dâu
- 8 → Không chắc liệu đó là Nho hay Dâu
- 9 → Nho
- 10 → Không chắc liệu đó là Táo hay Nho

Nếu dự đoán của bạn khớp với các dự đoán ở trên, bạn đã sử dụng thuật toán kNN rồi đó! Tại sao lại như vậy? Hãy xem xét kỹ hơn cách bạn đã thực hiện các dự đoán.

Trong ảnh, chúng ta quan sát được rằng các loại trái cây tương tự được sắp xếp cùng nhau. Đối với 1, 2 và 3, chúng ta có thể dễ dàng phân loại chúng là Cam vì chúng được bao quanh

chặt chẽ bởi Cam mà không có loại nào khác và do đó có khả năng cao rằng những quả ẩn bên dưới cũng có thể là Cam. Để nói một cách khác, những quả ẩn sẽ chủ yếu là cùng loại với đa số của hàng xóm của chúng. Cùng một lý do áp dụng cho 5, 6, 7 và 9.

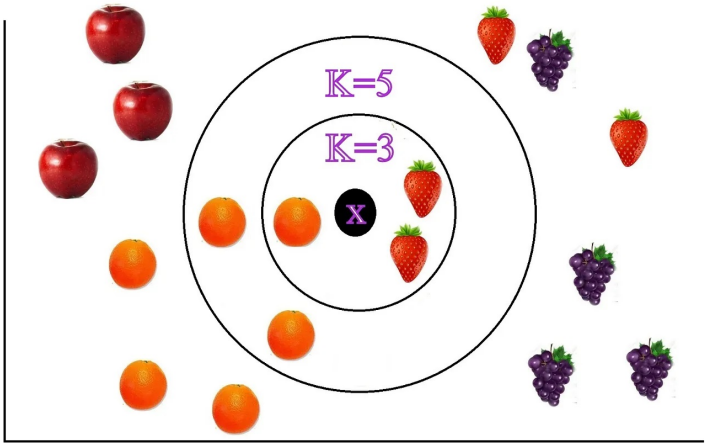
Đối với 10, chúng ta không chắc liệu đó là một quả Táo hay một quả Nho. Điều này là do, nó được bao quanh bởi cả Táo và Nho. Hoặc chúng ta có thể nói rằng hàng xóm của 10 thuộc về cả Táo và Nho. Vì vậy 10 có thể là một quả Táo hoặc một quả Nho. Tương tự cho 4 và 8.

Tóm lại, thuật toán kNN dự đoán nhãn cho một điểm dữ liệu mới dựa trên nhãn của các hàng xóm của nó. kNN dựa vào giả định rằng các điểm dữ liệu tương tự sẽ nằm gần nhau trong các tọa độ không gian. Trong ví dụ trên, dựa vào nhãn (Táo, Cam, Dâu, Nho) của các hàng xóm, chúng ta có thể dự đoán nhãn cho một điểm dữ liệu mới (quả trái cây ẩn).

3.2.2 Chỉ số k

Chỉ số k trong kNN là số lượng hàng xóm gần nhất mà chúng ta xem xét để thực hiện dự đoán. Chúng ta xác định sự gần nhau của một điểm dựa trên khoảng cách của nó (ví dụ: Euclidean, Manhattan, vv) với các điểm đang xem xét. Ví dụ, nếu $k = 5$, chúng ta xem xét 5 điểm gần nhất và lấy nhãn của đa số trong 5 điểm này làm nhãn được dự đoán.

Để làm rõ hơn, hãy xem xét cách xác định hàng xóm trong cửa hàng của Bob theo Hình 3. Mục tiêu của chúng ta là dự đoán nhãn cho điểm được đánh dấu là X. Nếu $k = 3$, trong 3 điểm hàng xóm của X, có 2 điểm là dâu và 1 điểm là cam. Vì vậy, chúng ta dự đoán nhãn cho X là dâu. Nếu K = 5, trong 5 điểm hàng xóm của X, có 3 điểm là cam và 2 điểm là dâu. Vì vậy, chúng ta dự đoán nhãn cho X là Cam.



Hình 3: Cách xác định nhãn của trái cây dựa vào chỉ số k

3.2.3 Giải thuật

Chúng ta có thể tạo ra một mô hình phân loại kNN bằng cách tuân theo các bước sau:

1. Tải dữ liệu.
2. Khởi tạo giá trị của k.
3. Để có được lớp được dự đoán, lặp từ 1 đến tổng số điểm dữ liệu huấn luyện.
4. Tính toán khoảng cách giữa dữ liệu kiểm tra và mỗi hàng của dữ liệu huấn luyện. Ở đây, chúng ta sẽ sử dụng khoảng cách Euclidean làm phương pháp đo khoảng cách vì nó là phương pháp phổ biến nhất.
5. Sắp xếp các khoảng cách tính được theo thứ tự tăng dần dựa trên giá trị khoảng cách.
6. Lấy k hàng đầu từ mảng đã sắp xếp.
7. Lấy lớp phổ biến nhất của các hàng này.
8. Trả về lớp được dự đoán.

3.3 Khoảng cách Euclidean

Khoảng cách Euclidean là một phương pháp đo khoảng cách giữa hai điểm trong không gian đa chiều. Nó được tính bằng cách lấy căn bậc hai của tổng bình phương của hiệu giữa các tọa độ của hai điểm.

Trong bài tập lớn này, để tính khoảng cách giữa các đặc trưng của một ảnh, gọi $x = \{x_1, x_2, ..., x_k, ..., x_n\}$ là vector chứa đặc trưng của ảnh 1 với n đặc trưng (hay có thể hiểu là số cột không phải nhãn của bảng dữ liệu), $y = \{y_1, y_2, ..., y_k, ..., y_n\}$ là vector chứa đặc trưng của anh 2 với n đặc trưng. Khoảng cách giữa 2 ảnh được tính theo công thức:

$$distance = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

3.4 Quá trình huấn luyện và dự đoán

Trong bài toán phân loại, quá trình huấn luyện và dự đoán là hai giai đoạn quan trọng trong việc xây dựng và triển khai công cụ dự đoán. Trong quá trình huấn luyện, công cụ được "huấn luyện" thông qua việc sử dụng dữ liệu huấn luyện để học các mối quan hệ trong dữ liệu. Quá trình này thường bao gồm việc chia dữ liệu thành các đặc trưng (features) và nhãn (labels). Trong bài tập lớn này, thông tin về các điểm ảnh sẽ là các đặc trưng và nhãn chính là con số thể hiện qua bức ảnh.

Sau khi đã chia dữ liệu thành đặc trưng và nhãn, chúng ta thường tiến hành chia dữ liệu thành hai tập con: tập huấn luyện (training set) và tập kiểm tra (test set).

- X_train và y_train: Đây là tập hợp các đặc trưng (X_train) và nhãn (y_train) tương ứng được sử dụng để huấn luyện mô hình. Mô hình sẽ học từ dữ liệu trong tập huấn luyện để tạo ra các dự đoán chính xác hơn về dữ liệu mới.
- X_test và y_test: Đây là tập hợp các đặc trưng (X_test) và nhãn (y_test) tương ứng được sử dụng để kiểm tra hiệu suất của mô hình sau khi đã huấn luyện. Dữ liệu trong tập kiểm tra được sử dụng để đánh giá xem mô hình có thể tổng quát hóa tốt ra những dữ liệu mới không.

Tóm lại, công cụ sẽ học từ X_train và y_train. Sau đó, tiến hành dự đoán trên X_test. Kết quả dự đoán sẽ đem so sánh với y_test để xem xét độ hiệu quả của việc dự đoán. Sinh viên có thể tham khảo testcase mẫu để thấy rõ quá trình huấn luyện và dự đoán.

4 Yêu cầu

Trong bài tập lớn này, sinh viên được yêu cầu phải thực hiện các class bên dưới để mô tả quá trình xử lí dữ liệu và hiện thực thuật toán k-nearest neighbor sử dụng cấu trúc danh sách, bao gồm: class List, class Dataset, class kNN.

4.1 class List

Class List là class dùng để lưu trữ dữ liệu theo cấu trúc danh sách và cung cấp cho người dùng các công cụ để xử lí danh sách. Lưu ý rằng, class List là một template class và chỉ là một interface. Sinh viên hãy kế thừa class thành một class khác này và override tất cả pure virtual method được liệt kê ở trên. Sinh viên được khuyến khích tham khảo chủ đề Inheritance, Polymorphism và Template trong C++.

Mô tả cho class List sẽ như sau:

```
template<typename T>
class List {
public:
    virtual ~List() = default;
    virtual void push_back(T value) = 0;
    virtual void push_front(T value) = 0;
    virtual void insert(int index, T value) = 0;
```

	SLL<int>	Array<int>
push_back	O(1)	O(1)
push_front	O(1)	[n*O(1)+O(n)]/(n+1) = O(1)
insert	O(n)	[n*O(1)+O(n)]/(n+1) = O(1)
remove	O(n)	[n*O(1)+O(n)]/(n+1) = O(1)
get	O(n)	O(1)
length	O(1)	O(1)
clear	O(n)	O(n)
print	O(n)	O(n)
reverse	O(n)	O(n)

```
    virtual void remove(int index) = 0;
    virtual T& get(int index) const = 0;
    virtual int length() const = 0 ;
    virtual void clear() = 0;
    virtual void print() const = 0;
    virtual void reverse() = 0;
};
```

Dưới đây là thông tin chi tiết về các phương thức trong class List:

1. `virtual ~List() = default;`
    - Virtual destructor cho class List.
2. `virtual void push_back(T value) = 0;`
    - Thêm một phần tử có giá trị `value` vào cuối danh sách
    - Ngoại lệ: Không.
3. `virtual void push_front(T value) = 0;`
    - Thêm một phần tử có giá trị *value* vào đầu danh sách
    - Ngoại lệ: Không.
4. `virtual void insert(int index, T value) = 0;`
    - Thêm một phần tử có giá trị `value` vào vị trí `index` của danh sách. Nếu index == kích thước hiện tại của danh sách (từ giờ gọi là `size`), thêm phần tử vào cuối danh sách
    - Ngoại lệ: Nếu `index` $<$ 0 hoặc `index` $>$ `size`, phương thức không làm gì cả.
5. `virtual void remove(int index) = 0;`
    - Xóa phần tử tại vị trí `index` của danh sách
    - Ngoại lệ: Nếu `index` $<$ 0 hoặc `index` $>=$ `size`, phương thức không làm gì cả.
6. `virtual T& get(int index) const = 0;`
    - Truy xuất phần tử tại vị trí `index` của danh sách. Trả về giá trị tham khảo đến phần tử đó
    - Ngoại lệ: Nếu `index` $<$ 0 hoặc `index` $>=$ `size`,
      thực thi câu lệnh `throw std::out_of_range("get(): Out of range")`.
7. `virtual int length() const = 0;`
    - Trả về độ dài hiện tại của danh sách
    - Ngoại lệ: Không.

8. `virtual void clear() = 0;`
    - Xóa tất cả phần tử trong danh sách, đưa danh sách về trạng thái khởi tạo ban đầu.
    - Ngoại lệ: Không.
9. `virtual void print() const = 0;`
    - In ra các phần tử trong danh sách theo format: các phần tử cách nhau một khoảng trắng, không có khoảng trắng dư ở cuối dòng.
    - Ngoại lệ: Không.
10. `virtual void reverse() = 0;`
    - Đảo ngược các phần tử trong danh sách một cách trực tiếp (in-place).
    - Ngoại lệ: Không.

	SLL<int>	Array<int>
push_back	O(1)	O(1)
push_front	O(1)	[n*O(1)+O(n)]/(n+1)
insert	O(n)	[n*O(1)+O(n)]/(n+1)
remove	O(n)	[n*O(1)+O(n)]/(n+1)
get	O(n)	O(1)
length	O(1)	O(1)
clear	O(n)	O(n)
print	O(n)	O(n)
reverse	O(n)	O(n)

4.2 class Dataset

Class Dataset là class dùng để lưu trữ và xử lí dữ liệu dạng bảng, cụ thể trong bài tập lớn này là bộ dữ liệu MNIST. Mô tả cho class Dataset sẽ như sau:

```
class Dataset {
private:
    List<List<int>*>* data; List<List<int>*>* = ArrayList<Image*>*
public:
    Dataset();
    ~Dataset();
    Dataset(const Dataset& other);
    Dataset& operator=(const Dataset& other);
    bool loadFromCSV(const char* fileName);
    void printHead(int nRows = 5, int nCols = 5) const;
    void printTail(int nRows = 5, int nCols = 5) const;
    void getShape(int& nRows, int& nCols) const;
    void columns() const;
    bool drop(int axis = 0, int index = 0, std::string columns = "");
    Dataset extract(int startRow = 0, int endRow = -1, int startCol = 0, int
    endCol = -1) const;
};
```

Dưới đây là thông tin chi tiết về các trường dữ liệu và phương thức trong class Dataset:

1. `List<List<int>*>* data;`
    - Là trường thông tin chính để lưu giữ dữ liệu ở dạng bảng.

2. `Dataset();`
    - Default constructor.
3. `~Dataset();`
    - Destructor.
4. `Dataset(const Dataset& other);`
    - Copy constructor.
5. `Dataset& operator=(const Dataset& other);`
    - Assignment operator.
6. `bool loadFromCSV(const char* fileName);`
    - Phương thức được sử dụng để tải dữ liệu từ file `fileName`, cụ thể trong bài tập lớn này là file `mnist.csv`. Các thông tin trong file sẽ được lưu trữ vào biến `data` và các b iến khác do sinh viên đề xuất.
    - Hàm trả về true nếu việc tải dữ liệu thành công, ngược lại false.
7. `void printHead(int nRows = 5, int nCols = 5) const;`
    - In ra `nRows` dòng đầu tiên, và chỉ in `nCols` cột đầu tiên của bảng dữ liệu.
    - Format in:
        - Dòng đầu tiên, in ra tên các cột của bảng dữ liệu.
        - Từ dòng thứ 2 trở đi, in giá trị của mỗi ô trong bảng, mỗi phần tử cách nhau bằng khoảng trắng, không có khoảng trắng dư ở cuối dòng.
    - Ngoại lệ: Nếu `nRows` lớn hơn số lượng dòng trong bảng dữ liệu, in hết dòng trong bảng dữ liệu. Nếu `nCols` lớn hơn số lượng cột trong bảng dữ liệu, in hết cột trong bảng dữ liệu. Nếu `nRows` hoặc `nCols` nhỏ hơn 0, không in gì cả.

Ví dụ 4.1

Kết quả in ra của `printHead` với `nRows` = 5 và `nCols` = 5, với dữ liệu từ file `mnist.csv`:

label 1x1 1x2 1x3 1x4
5 0 0 0 0
0 0 0 0 0
4 0 0 0 0
1 0 0 0 0
9 0 0 0 0

8. `void printTail(int nRows = 5, int nCols = 5) const;`
    - In ra `nRows` dòng cuối cùng, và chỉ in `nCols` cột cuối cùng của bảng dữ liệu.
    - Format in:
        - Dòng đầu tiên, in ra tên các cột của bảng dữ liệu.
        - Từ dòng thứ 2 trở đi, in giá trị của mỗi ô trong bảng, mỗi phần tử cách nhau bằng khoảng trắng, không có khoảng trắng dư ở cuối dòng.
    - Ngoại lệ: Nếu `nRows` lớn hơn số lượng dòng trong bảng dữ liệu, in hết dòng trong bảng dữ liệu. Nếu `nCols` lớn hơn số lượng cột trong bảng dữ liệu, in hết cột trong bảng dữ liệu. Nếu `nRows` hoặc `nCols` nhỏ hơn 0, không in gì cả.

    Ví dụ 4.2

    Kết quả in ra của `printTail` với `nRows` = 5 và `nCols` = 5, với dữ liệu từ file `mnist.csv`:

    28x24 28x25 28x26 28x27 28x28
    0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0

9. `void getShape(int& nRows, int& nCols) const;`
    - Phương thức trả về số lượng dòng và số lượng cột tổng cộng hiện tại của đối tượng thông qua hai tham số tham khảo `nRows` và `nCols`.
10. `void columns() const;`
    - In ra tên của tất cả các cột của bảng dữ liệu. Mỗi tên cách nhau một khoảng trắng, không có khoảng trắng dư cuối cùng.
11. `bool Dataset::drop(int axis = 0, int index = 0, std::string columnName = "");`
    - Xóa một hàng hoặc cột của bảng dữ liệu.
    - Nếu `axis` khác 0 hoặc 1, hàm không làm gì cả và trả về false.
    - Nếu `axis` == 0, thực hiện xóa một hàng tại vị trí `index`, sau đó trả về true. Nếu index >= số dòng hoặc < 0, hàm không làm gì cả và trả về false.
    - Nếu `axis` == 1, thực hiện xóa một cột có tên trùng với `columnName`, sau đó trả về true. Nếu columnName không nằm trong danh sách tên các cột, hàm không làm gì cả và trả về false.

12. `Dataset extract(int startRow = 0, int endRow = -1, int startCol = 0, int endCol = -1) const;`
    - Phương thức dùng để trích xuất một phần của bảng dữ liệu, sau đó trả về bảng dữ liệu đã trích xuất.
    - Trong đó: `startRow` là hàng bắt đầu, `endRow` là hàng cuối cùng, `startCol` là cột bắt đầu, `endCol` là cột cuối cùng.
    - Nếu endRow == -1, ta sẽ lấy tất cả các hàng. Tương tự với endCol == -1.
    - Các testcase sẽ đảm bảo startRow, endRow, startCol và endCol nằm trong khoảng giá trị hợp lệ (từ 0 đến số hàng/số cột) và start $\leq$ end.

Vì việc chỉnh sửa dữ liệu sẽ thực thi nhiều lần, liệu bạn có thể hiện thực thao tác thêm/xóa trên danh sách với độ phức tạp O(1)?

4.3 class kNN

Class kNN là class dùng để hiện thực các chức năng của mô hình dự đoán, sử dụng thuật toán kNN được trình bày ở các mục phía trên. Thiết kế cho class kNN sẽ như sau:

```
class kNN {
private:
    int k;
public:
    kNN(int k = 5);
    void fit(const Dataset& X_train, const Dataset& y_train);
    Dataset predict(const Dataset& X_test);
    double score(const Dataset& y_test, const Dataset& y_pred);
};
```

Dưới đây là thông tin chi tiết về các trường dữ liệu và phương thức trong class kNN:

1. `int k;`
    - Chỉ số k, đã được mô tả ở mục trên.
2. `void fit(const Dataset& X_train, const Dataset& y_train);`
    - Thực hiện tải dữ liệu để huấn luyện, trong đó:
        - X_train: Bảng dữ liệu chứa các đặc trưng không phải là nhãn. Số hàng của X_train là số lượng ảnh được sử dụng để huấn luyện, và số cột là số đặc trưng.
        - y_train: Bảng dữ liệu chứa các nhãn tương ứng với các đặc trưng. Vì chỉ là các nhãn, nên số hàng của y_train là số lượng ảnh, và số cột chỉ duy nhất là 1.

3. `Dataset predict(const Dataset& X_test);`
    - Là phương thức thực thi việc dự đoán dựa vào thuật toán kNN.
    - X_test là bảng dữ liệu chứa đặc trưng của các ảnh đem dự đoán. Số hàng của X_test là số lượng ảnh được sử dụng để dự đoán, và số cột là số đặc trưng.
    - Lưu ý rằng, giải thuật kNN đã mô tả ở mục trên chỉ áp dụng cho dự đoán 1 ảnh mới, sinh viên cần phải hiện thực dự đoán cho nhiều ảnh mới, tương ứng với số hàng trong bảng dữ liệu X_test.
    - Kết quả trả về là bảng dữ liệu gồm nhiều hàng và 1 cột. Trong đó, số hàng là số ảnh được đem dự đoán. Nội dung của từng ô trong cột là nhãn (lớp) sau khi được dự đoán bởi thuật toán kNN.
    - Vì việc truy xuất vào bảng dữ liệu là diễn ra thường xuyên. Liệu có cách nào hiện thực thao tác truy xuất với độ phức tạp O(1)?
4. `double score(const Dataset& y_test, const Dataset& y_pred);`
    - Là phương thức đo đạc độ chính xác của quá trình dự đoán.
    - y_test là bảng dữ liệu chứa nhãn thực sự các ảnh đem dự đoán. Số hàng của y_test là số lượng ảnh được sử dụng để dự đoán, và số cột là 1.
    - y_pred là bảng dữ liệu chứa nhãn sau khi dự đoán bởi giải thuật kNN (thu được từ hàm predict). Số hàng của y_pred là số lượng ảnh được sử dụng để dự đoán, và số cột là 1.
    - Kết quả trả về là một số thực với cách tính như sau:

      $$\text{Độ chính xác} = \frac{\text{Số ảnh dự đoán đúng}}{\text{Tổng ảnh dự đoán}}$$

      Trong đó, Số ảnh dự đoán đúng là số ảnh có nhãn dữ đoán trùng với nhãn thực sự.

4.4 Hàm train_test_split

Đây là hàm phụ trợ dùng để chia bộ dữ liệu ra thành các bảng dữ liệu nhỏ hơn phục vụ cho quá trình huấn luyện và dự đoán. Prototype của hàm như sau:

```
void train_test_split(Dataset& X, Dataset& y, double test_size, Dataset&
    X_train, Dataset& X_test, Dataset& y_train, Dataset& y_test);
```

Trong đó:

- Input:
    - Dataset& X: Bảng dữ liệu chứa các đặc trưng của ảnh. Số hàng là số lượng ảnh, Số cột là số lượng các đặc trưng.

  - Dataset& y: Bảng dữ liệu chứa nhãn của ảnh. Số hàng là số lượng ảnh, Số cột là 1.
  - double test_size: Số thực xác định tỉ lệ của tập huấn luyện trên tổng thể dữ liệu. Phần còn lại sẽ là tập dự đoán.

- Output:
  - Dataset& X_train: Bảng dữ liệu huấn luyện chứa các đặc trưng của ảnh. Số hàng là số lượng ảnh huấn luyện, Số cột là số lượng các đặc trưng.
  - Dataset& y_train: Bảng dữ liệu huấn luyện chứa nhãn của ảnh. Số hàng là số lượng ảnh huấn luyện, Số cột là 1.
  - Dataset& X_test: Bảng dữ liệu dự đoán chứa các đặc trưng của ảnh. Số hàng là số lượng ảnh, Số cột là số lượng các đặc trưng.
  - Dataset& y_test: Bảng dữ liệu dự đoán chứa nhãn thực sự của ảnh. Số hàng là số lượng ảnh, Số cột là 1.

4.5 Hướng dẫn

Để hoàn thành bài tập lớn này, sinh viên phải:

1. Đọc toàn bộ tập tin mô tả này.
2. Tải xuống tập tin initial.zip và giải nén nó. Sau khi giải nén, sinh viên sẽ nhận được các tập tin: main.cpp, main.hpp, kNN.hpp, kNN.cpp và file mnist.csv. Sinh viên sẽ chỉ nộp 2 tập tin là kNN.hpp và kNN.cpp nên không được sửa đổi tập tin main.h khi chạy thử chương trình.
3. Sinh viên sử dụng câu lệnh sau để biên dịch:
   g++ -o main main.cpp kNN.cpp -I . -std=c++11
   Câu lệnh trên được dùng trong command prompt/terminal để biên dịch chương trình. Nếu sinh viên dùng IDE để chạy chương trình, sinh viên cần chú ý: thêm đầy đủ các tập tin vào project/workspace của IDE; thay đổi lệnh biên dịch của IDE cho phù hợp. IDE thường cung cấp các nút (button) cho việc biên dịch (Build) và chạy chương trình (Run). Khi nhấn Build IDE sẽ chạy một câu lệnh biên dịch tương ứng, thông thường, chỉ biên dịch phải main.cpp. Sinh viên cần tìm cách cấu hình trên IDE để thay đổi lệnh biên dịch: thêm file kNN.cpp, thêm option -std=c++11, -I .
4. Chương trình sẽ được chấm trên nền tảng UNIX. Nền tảng và trình biên dịch của sinh viên có thể khác với nơi chấm thực tế. Nơi nộp bài trên BKeL được cài đặt giống với nơi chấm thực tế. Sinh viên phải chạy thử chương trình trên nơi nộp bài và phải sửa tất cả các lỗi xảy ra ở nơi nộp bài BKeL để có đúng kết quả khi chấm thực tế.

5. Sửa đổi các file kNN.hpp, kNN.cpp để hoàn thành bài tập lớn này và đảm bảo hai yêu cầu sau:
    - Tất cả các phương thức trong mô tả này đều phải được hiện thực để việc biên dịch được thực hiện thành công. Nếu sinh viên chưa thể hiện thực được phương thức nào, hãy cung cấp một hiện thực rỗng cho phương thức đó. Mỗi testcase sẽ gọi một số phương thức đã mô tả để kiểm tra kết quả trả về.
    - Chỉ có 1 lệnh **include** trong tập tin kNN.hpp là **#include "main.hpp"** và một include trong tập tin kNN.cpp là **#include "kNN.hpp"**. Ngoài ra, không cho phép có một **#include** nào khác trong các tập tin này.
6. Trong tập tin main.cpp có cung cấp một số testcases đơn giản trong các hàm có định dạng **"tc<x>()"**.
7. Sinh viên được phép viết thêm các phương thức và thuộc tính khác trong các class được yêu cầu hiện thực. Sinh viên được phép viết thêm các class khác ngoài các yêu cầu hiện thực.
8. Sinh viên được yêu cầu thiết kế và sử dụng các cấu trúc dữ liệu dựa trên các loại danh sách đã học.
9. Sinh viên phải giải phóng toàn bộ vùng nhớ đã xin cấp phát động khi chương trình kết thúc.

5 Nộp bài

Sinh viên chỉ nộp 2 tập tin: kNN.hpp và kNN.cpp, trước thời hạn được đưa ra trong đường dẫn "Assignment 1 - Submission". Có một số testcase đơn giản được sử dụng để kiểm tra bài làm của sinh viên nhằm đảm bảo rằng kết quả của sinh viên có thể biên dịch và chạy được. Sinh viên có thể nộp bài bao nhiêu lần tùy ý nhưng chỉ có bài nộp cuối cùng được tính điểm. Vì hệ thống không thể chịu tải khi quá nhiều sinh viên nộp bài cùng một lúc, vì vậy sinh viên nên nộp bài càng sớm càng tốt. Sinh viên sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên sinh viên sẽ không thể nộp nữa. Bài nộp qua các phương thức khác đều không được chấp nhận.

6 Một số quy định khác

- Sinh viên phải tự mình hoàn thành bài tập lớn này và phải ngăn không cho người khác đánh cắp kết quả của mình. Nếu không, sinh viên sẽ bị xử lý theo quy định của trường

vì gian lận.

- Mọi quyết định của giảng viên phụ trách bài tập lớn là quyết định cuối cùng.
- Sinh viên không được cung cấp testcase sau khi chấm bài mà chỉ được cung cấp thông tin về chiến lược thiết kế testcase và phân bố số lượng sinh viên đúng theo từng testcase.
- Nội dung Bài tập lớn sẽ được Harmony với một câu hỏi trong bài kiểm tra với nội dung tương tự.

——————————HẾT——————————