ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN HỌC

NHẬN DẠNG CHỮ SỐ TRONG BỘ DỮ LIỆU MNIST SỬ DỤNG THUẬT TOÁN KNN

Giảng viên: PGS.TS. Lê Đình Duy
ThS. Phạm Nguyễn Trường An

Lớp : CS114.K21

Sinh viên thực hiện:

PHAN QUỐC CƯỜNG	16521639
BÙI THẾ DUY	16521604

TP. Hồ Chí Minh, ngày 04 tháng 08 năm 2020

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Đầu tiên, với tất cả lòng biết ơn và sự kính trọng, chúng em xin trân trọng cảm ơn quý thầy, cô khoa Khoa học máy tính, cũng như các thầy, cô đang công tác tại trường Đại học Công nghệ Thông tin – ĐHQG-HCM đã dùng tất cả tri thức và tâm huyết để truyền đạt những kiến thức, kinh nghiệm quý báu cho chúng em trong suốt quá trình học tập và rèn luyện tại ngôi trường này.

Đặc biệt, chúng em xin gửi lời tri ân chân thành và sâu sắc đến thầy Lê Đình Duy và thầy Phạm Nguyễn Trường An – những người thầy hướng dẫn đã hết sức tận tâm, nhiệt tình hỗ trợ và hết lòng giúp đỡ cho nhóm chúng em trong suốt quá trình thực hiện đề tài. Những định hướng, bổ sung, góp ý của thầy là nguồn cảm hứng và nền tảng cơ sở góp phần giúp chúng em có những nghiên cứu đúng đắn, đạt được kết quả tốt nhất trong việc xây dựng, phát triển ứng dụng và hoàn thiện đề tài.

Tiếp theo, chúng em xin được phép gửi lời cảm ơn đặc biệt đến gia đình và người thân. Gia đình luôn là chỗ dựa tinh thần vững chắc, là nguồn động lực lớn giúp chúng em vượt qua mọi khó khăn, phấn đấu hoàn thành tốt đề tài này.

Cuối cùng, nhóm chúng em xin gửi lời cảm ơn đến các anh, chị và các bạn sinh viên trường Đại học Công nghệ Thông tin đã nhiệt tình hỗ trợ, chia sẻ ý kiến, góp ý giúp chúng em trong suốt thời gian thực hiện đề tài.

Một lần nữa, chúng em xin chân thành cảm ơn và xin gửi lời chúc sức khỏe đến quý thầy, cô. Kính chúc khoa Khoa học máy tính ngày càng phát triển và thành công trong sự nghiệp “chèo đò” cao quý.

TP. Hồ Chí Minh, ngày 04 tháng 08 năm 2020

Nhóm tác giả

MỤC LỤC

MỞ ĐẦU

Nhận dạng chữ viết tay của các ký tự đã xuất hiện từ những năm 1980. Nhiệm vụ nhận dạng chữ số viết tay, sử dụng bộ phân loại, có tầm quan trọng và được sử dụng rất nhiều như - nhận dạng chữ viết tay trực tuyến trên máy tính bảng, nhận dạng mã zip trên thư để phân loại thư bưu điện, ngân hàng xử lý kiểm tra số tiền, mục nhập số trong các biểu mẫu điền bằng tay (ví dụ - biểu mẫu thuế), v.v. Có những thách thức khác nhau phải đối mặt khi cố gắng giải quyết vấn đề này. Các chữ số viết tay không phải lúc nào cũng có cùng kích thước, độ dày hoặc hướng và vị trí so với lề. Mục tiêu của tôi là triển khai một phương pháp phân loại theo mẫu để nhận ra các chữ số viết tay được cung cấp trong bộ dữ liệu MNIST về hình ảnh các chữ số viết tay (0-9).

Bài báo cáo gồm các phần:

- Cơ sở lý thuyết
- Phân tích thiết kế thuật toán
- Kết luận và hướng phát triển

Chương 1. **Cơ sở lý thuyết các thuật toán**

1.1. Giới thiệu về bộ dữ liệu MNIST

Modified National Institute of Standards and Technology database (viết tắt là MNIST) là bộ dữ liệu được xây dựng từ hai bộ dữ liệu NIST's Special Database 3 và Special Database 1.

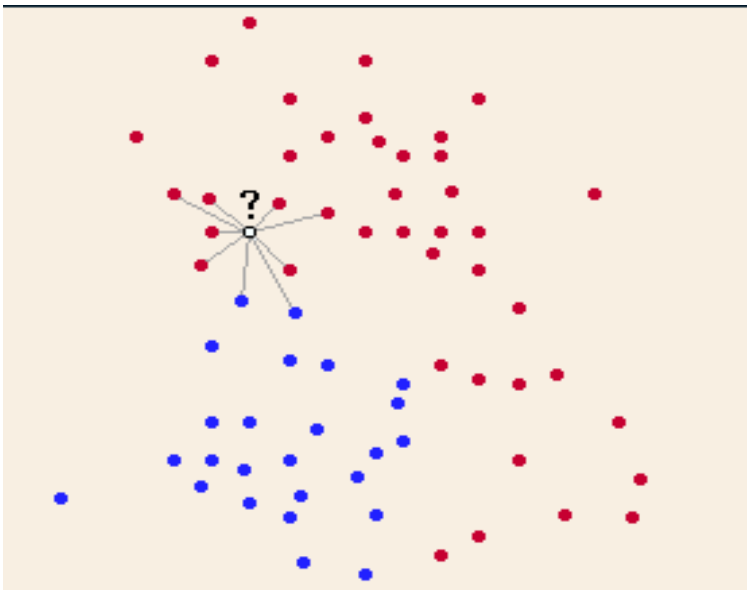
Hình 1. Ảnh bộ dữ liệu được trích ra từ MNIST

MNIST gồm 60000 hình cho training và 10000 hình cho testing lưu ở dạng ma trận kích thước 28×28 và range dao động từ 0−256, kiểu dữ liệu uint8. Mỗi hình đều có ground truth label tương ứng. Có tổng cộng 10 class trong MNIST tương ứng với các chữ số từ 0−9.

1.2. Thuật toán K-Nearest Neighbor

K-Nearest Neighbor (viết tắt là KNN) là một thuật toán của Machine Learning. KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của

K điểm dữ liệu trong training set gần nó nhất (K-lân cận). Hiểu đơn giản là tìm k phần tử giống phần tử được test nhất, kết quả là class nào xuất hiện nhiều nhất thì đó là kết quả cần tìm.



Hình 2. Ảnh minh họa thuật toán KNN

1.3. Các thư viện sử dụng

1.3.1. Numpy

Numpy (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng "core Python" đơn thuần.

Hình 3. Thư viện numpy

1.3.2. Pandas

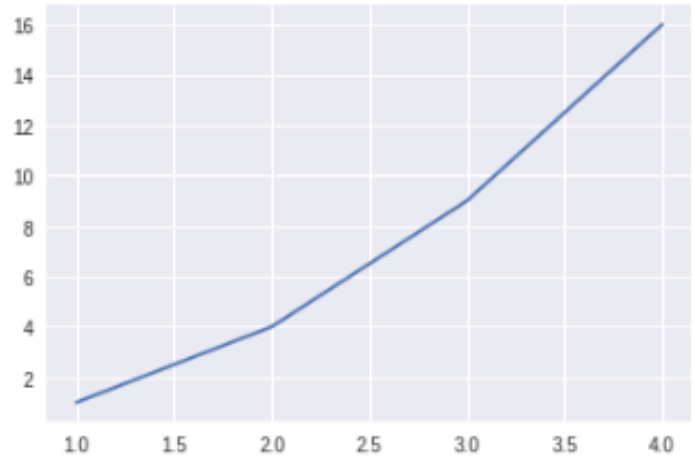
Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.

1.3.3. Matplotlib

Matploblib là một thư viện trực quan hoá dữ liệu phổ biến trong Python. Nó có thể vẽ được nhiều loại đồ thị khác nhau, và rất hữu ích khi làm việc cùng với NumPy.

```
import matplotlib.pyplot as plt
import numpy as np

plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



Hình 4. Ví dụ về Matplotlib

1.3.4. Scikit - Learn

Hình 5. Thư viện Scikit – Learn

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Chương 2. **Phân tích thiết kế chương trình**

2.1. Thiết kế giải thuật

B1: Chuẩn bị bộ dữ liệu ký tự

B2: Xử lý dữ liệu

B3: Training dữ liệu bằng thuật toán KNN

B4: Test (predict) kết quả

2.2. Hiện thực hóa giải thuật

2.2.1. Chuẩn bị bộ dữ liệu ký tự

Tập dữ liệu được sử dụng bao gồm 300 hình ảnh training và 300 hình ảnh testing là tập hợp con của bộ dữ liệu MNIST[1] (ban đầu bao gồm 60.000 hình ảnh training và 10.000 hình ảnh testing). Mỗi hình ảnh có kích thước 28 x 28 (0-255) được gán nhãn (label).

2.2.2. Xử lý dữ liệu

- Chuyển đổi hình ảnh mức xám thành hình ảnh nhị phân
- Convert hình ảnh nhị phân thanh mảng một chiều duy nhất [1,n]
- Gán nhãn (label) của mỗi mảng cùng với nó

2.2.3. Training dữ liệu bằng thuật toán KNN

Sử dụng thuật toán KNN classifier trong thư viện scikit – learn để training dữ liệu bằng tập dữ liệu train_data với trường hợp K = 5

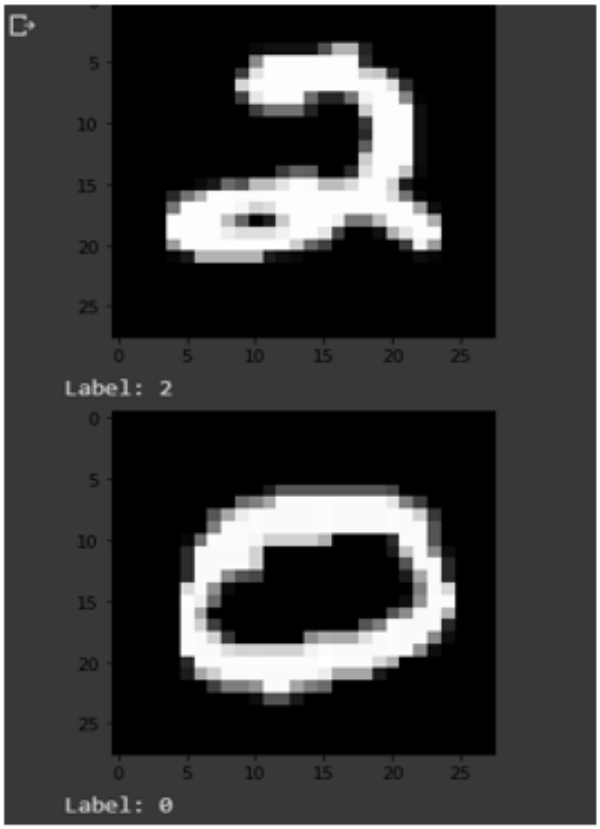
```
clf = neighbors.KNeighborsClassifier(n_neighbors = 5, p = 2, weights = 'distance')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Hình 6. Ảnh minh họa

2.2.4. Test (predict) kết quả

Sau khi training dữ liệu bằng thuật toán KNN chúng ta tiến hành test (predict) kết quả với tập dữ liệu test_data

```
[ ]  x=0
     for test in test_images:
         im = test.reshape((28,28))
         y_pred = clf.predict(test_images)
         plt.figure()
         plt.imshow(im, cmap='gray')
         plt.show()
         print ("Label:",y_pred[x])
         x=x+1
```



Hình 7. Kết Quả

Chương 3. **Kết luận và hướng phát triển**

3.1. Kết luận

3.1.1. Ưu điểm

- Dễ sử dụng và cài đặt
- Xử lý tốt với dữ liệu nhiễu (do dựa trên khoảng cách để quyết định phân lớp)

3.1.2. Nhược điểm

- Phụ thuộc vào giá trị k do người dùng lựa chọn. Nếu k quá nhỏ, nhạy cảm với nhiễu. Nếu k quá lớn, vùng lân cận có thể chứa các điểm của lớp khác dẫn đến kết quả không được chính xác
- Vấn đề chung mà sẽ gặp phải trong bài toán phân loại chữ số này là sự giống nhau giữa các chữ số như 1 và 7, 9 và 0, 3 và 8, 9 và 8,… Ngoài ra, người ta còn viết cùng một chữ số nhưng theo nhiều cách khác nhau.
- Thuật toán KNN không thực sự "học" bất cứ điều gì – nếu thuật toán mắc lỗi, nó không có cách nào để "sửa" và "cải thiện" chinh nó cho các phân loại sau này
- Tỉ lệ dự đoán chinh xác (accuracy) là 93,3% chưa thực sự tốt

3.2. Hướng phát triển

- Ứng dụng test real-time (opencv, svm,…)
- Thay thế dataset có kích thước lớn hơn
- Áp dụng các cấu trúc thuật toán khác hiệu quả hơn cho kết quả chính xác hơn

Tài liệu kham khảo:

https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz

https://www.pyimagesearch.com/category/machine-learning/