NHẬN DIỄN CHỮ SỐ VIẾT TAY

By
Group 4

Class DAMI330484_01
Instructor Quach Dinh Hoang
June 04th, 2022
Faculty of Information Technology
HCMC University of Technology and Education

ABSTRACT

Nhận dạng chữ viết tay (Handwriting recognition - HWR), còn được gọi là Nhận dạng văn bản viết tay (Handwritten Text Recognition - HTR), là khả năng máy tính nhận và giải thích dữ liệu chữ viết tay dễ hiểu từ các nguồn như tài liệu giấy, ảnh, màn hình cảm ứng và các thiết bị khác. Hình ảnh của văn bản viết có thể được cảm nhận "tắt dòng" từ một mảnh giấy bằng cách quét quang học (nhận dạng ký tự quang học) hoặc nhận dạng từ thông minh. Ngoài ra, các chuyển động của đầu bút có thể được cảm nhận "trên đường", ví dụ như bề mặt màn hình máy tính dựa trên bút, một công việc thường dễ dàng hơn vì có nhiều manh mối hơn. Một hệ thống nhận dạng chữ viết xử lý định dạng, thực hiện phân đoạn chính xác thành các ký tự và tìm các từ hợp lý nhất..

1. Introduction

Nhận dạng chữ số viết tay là khả năng máy tính nhận dạng chữ số viết tay của con người. Đó là một nhiệm vụ khó khăn cho cỗ máy vì các chữ số viết tay không hoàn hảo và có thể được tạo ra với nhiều kiểu cách khác nhau. Nhận dạng chữ số viết tay là giải pháp cho vấn đề này sử dụng hình ảnh của một chữ số và nhận dạng chữ số hiện diện trong hình ảnh.

2. Data

Bài toán sẽ thực hiện phân tích sẽ dựa trên tập dữ liệu trong một cuộc thi của kaggle. Các tệp dữ liệu train.csv và test.csv chứa hình ảnh thang màu xám của các chữ số vẽ tay, từ 0 đến 9..

Mỗi hình ảnh có chiều cao 28 pixel và chiều rộng 28 pixel, tổng cộng là 784 pixel. Mỗi pixel có một giá trị pixel duy nhất được liên kết với nó, cho biết độ sáng hoặc tối của pixel đó, với các con số cao hơn có nghĩa là tối hơn. Giá trị pixel này là một số nguyên từ 0 đến 255

Tập train.csv có 785 cột. Trong đó cột đầu tiên là nhãn và phần còn lại của các cột chứa các giá trị pixel của hình ảnh được liên kết.

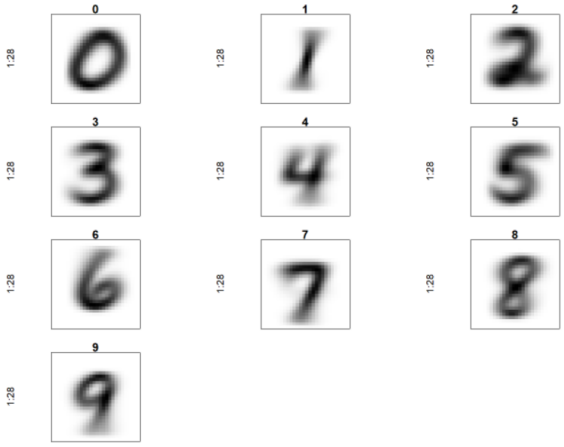
2.1. Transform

Tập dữ liệu có 2 file đó là train.csv là test.csv, tập train chưa 785 cột với cột đầu tiên là label là nhãn tương ứng với các chữ số viết tay và các giá trị pixel sẽ có giá trị từ 0 đến 255 với các pixel có giá trị càng lớn thì có nghĩa là nó càng tối

Ví dụ: pixel31 cho biết pixel nằm trong cột thứ tư từ bên trái và hàng thứ hai từ trên cùng, như trong biểu đồ bên dưới.

```
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
 |   |   |   | ...  |   |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
```

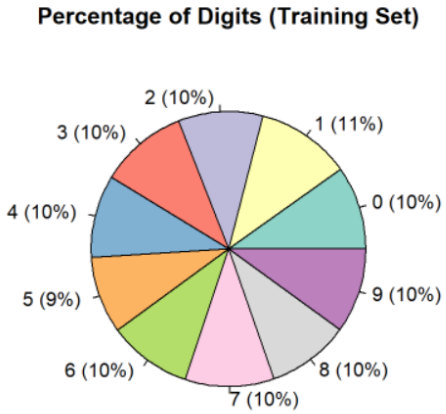
3. DATA VISUALIZATION



Các chữ số trực quan lên trong khung ảnh 28 pixel chiều rộng và 28 pixel chiều cao thì các chữ số khá là mờ không rõ ở nhiều chỗ, việc này có thể ảnh hưởng rất nhiều đến việc dự đoán của chúng ta.

2.2. Missing Value

Trong tập dữ liệu các cột của tập dữ liệu đều đóng một vai trò quan trọng trong việc nhận diện các nhãn thật sự của nó nên chúng ta không thể loại bỏ bất kì cột thuộc tính nào cùng như các giá trị không có giá trị nào bị NA hoặc la NULL.



Trong thành phần của tập train tương đối cân bằng với mỗi nhãn đóng góp 10% trong toàn bộ tập dữ liệu. Điều này giúp chúng ta có thể dự đoán tốt hơn và sử dụng được độ đo phù hợp với tập dữ liệu này.

4. Phương pháp

Sau khi xem phân tích dữ, ta thấy phân bố của chúng khá tương đối đều nên chúng ta không cần căn chỉnh đối với tập dữ liệu này.

Các mô hình sẽ được sử dụng với đề tài này lần lượt là: K nearest neighbor, Random Forest, Decision Tree. Đối với tập

dữ liệu cân bằng này và cũng như yêu cầu của cuộc thi thì chúng ta lựa chọn độ đo Accuracy để đánh giá thuật toán.

****Thuật toán K nearest neighbor:**

Thuật toán K nearest neighbor, hay còn được gọi là K-NN hoặc là KNN, là một thuật toán học có giám sát, đơn giản và hiệu quả trong học máy.

KNN hoạt động với nguyên lý tương tự. Ý tưởng của thuật toán KNN đó là tìm ra output của một điểm dữ liệu dựa vào output của K điểm gần nhất xung quanh nó. Trong thực tế, để đo khoảng cách giữa các điểm dữ liệu thì chúng ta có thể sử dụng rất nhiều độ đo. tiêu biểu là Mahatan, Euclid, Cosin,..

Hàm Minkowski:

$$d(x,z) = \left(\sum_{i=1}^{n} |x_i - z_i|^p \right)^{1/p}$$

Hàm Manhattan:

$$d(x,z) = \sum_{i=1}^{n} |x_i - z_i|$$

Hàm Euclid:

$$d(x,z) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2}$$

Giả sử ta có k=3, điều này có nghĩa là dựa vào 3 điểm gần nhất với một độ đo nào để để xác output của một điểm dữ liệu nào đó.

***Các bước trong KNN

- Chúng ta cần phải xác định tham số K(số điểm gần nhất)
- Tính khoảng cách bằng các độ đo với tất cả các đối tượng.
- Sắp xếp khoảng cách đó theo thứ tự tăng dần và chọn ra K điểm gần nhất.
- Lấy ra K điểm gần nhất đã được xác định ở trên
- Lấy ra lớp dữ liệu xuất hiện nhiều nhất
- Lớp của dữ liệu với sẽ dựa vào lớp mà ta thực hiện ở trên,

****Thuật toán Decision Tree**

Decision tree là một thuật toán học có giám sát và có thể sử dụng vào hai bài toán phân loại và tuyến tính.

- Cây quyết định với bài toán phân loại: biến đầu ra sẽ là biến phân loại, các nhãn, các lớp
- Cây quyết định với bài toán hồi quy: trong bài toán này thì biến đâu ra sẽ là biến liên tục.

Các độ đo độ sách của các node trong Decision tree:

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Với pi là tần suất lớp i tại node t và c là tổng số lớp

- Entropy

Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Phân loại sai

$$Classification\ error = 1 - \max[p_i(t)]$$

- Information Gain

$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

Node cha, p được chia thành k các phân vùng con, ni là số record trong node thứ i, và n là tổng số record node cha.

****Thuật toán Random Forest**

Random Forest là một thuật toán học có giám sát, cũng như Decision tree nó sử dụng được cả cho bài toán hồi quy và bài toán phân loại. Random Forest là một thuật toán linh hoạt và tương đối dễ sử dụng.

Thuật toán Random Forest kết hợp nhiều cây quyết định để xác định đầu ra cuối cùng thay vì lựa chọn dựa vào các cây riêng lẻ.

Random Forest hoạt động bằng cách lấy ngẫu nhiên d thuộc tính từ tập dữ liệu tạo ra cây và sử dụng kết hợp các cây quyết định đó để đưa ra kết quả cuối cùng.

Để hiểu rõ hơn về hoạt động của Random Forest thì ta có các bước sau

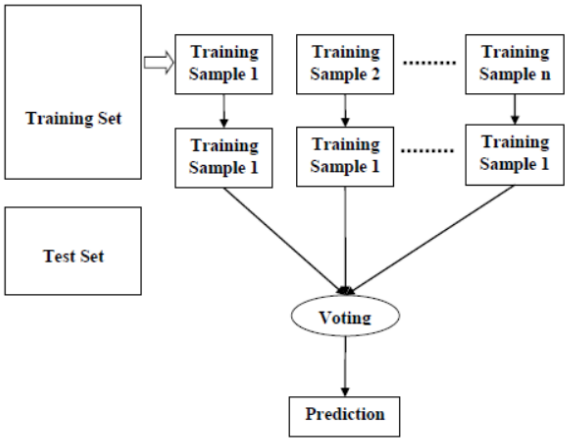
Bước 1: Chọn mẫu ngẫu nhiên tự một tập dữ liệu nhất định

Bước 2: Với các mẫu ngẫu nhiên thì thuật toán sẽ xây dựng từng cây quyết định với từng mẫu và nó kết quả sẽ từ các cây quyết định này.

Bước 3: Dựa vào kết quả của các cây quyết định thì chúng ta sẽ thực hiện việc voting(hiểu như là việc bỏ phiếu)

Bước 4: Dựa vào kết quả voting bên trên thì ta chọn voting nhiều nhất làm kết quả cuối cùng.

Có một điểm chú ý là số thuộc tính ta lấy ngẫu nhiên từ tập dữ liệu, việc lấy số thuộc tính này sẽ quyết định đến kết quả của bài toán.



5. BUILD MODEL

Để xây dựng các mô hình phân loại trên tập dữ liệu thì chúng chọn ra 10% tập dữ liệu để chọn ra siêu tham số với từng mô hình. Với 10% dữ liệu đó ta chọn 70% để train và 30% cho đánh giá thuật toán.

Thuật toán Knn ta cho giá trị k chạy từ 20 đến 29 và được kết quả như sau:

Với k = 26 ta được

```
 Accuracy
0.9092857
```

Với k = 27

```
 Accuracy
0.9121429
```

Với k = 28

```
 Accuracy
0.8921429
```

K cho độ chính xác cao nhất là k =27 với 0.912. Tuy nhiên độ chính xác của các giá trị k không có sự chênh lệch quá nhiều.

Ta sử dụng K = 27 để thực hiện dự đoán nhãn trên tập test.csv.

Kết quả mà ta thu được khi submit ở cuộc thi trên Kaggle là:

0.95450

Với kết quả này thì chúng ta được độ chính xác cao khoảng 95.45%.

Mô hình Random Forest:

Với mô hình random forest ta thực hiện chọn siêu tham số với 10% dữ liệu được trích ra, ta được các kết quả như sau:

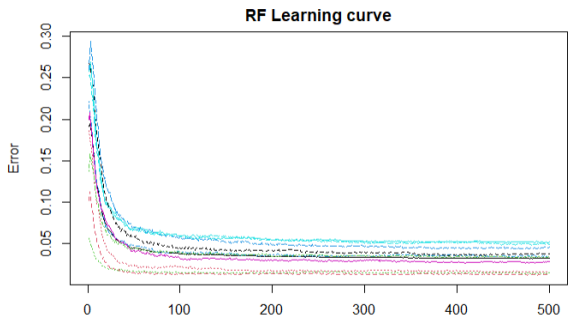
Với ntree = 150, mtry = 28 thì ta được ma trận nhầm lẫn như sau

```
Call:
 randomForest(x = trainSet[, -785], y = trainSet[, 785],
               Type of random forest: classification
                     Number of trees: 150
No. of variables tried at each split: 28

        OOB estimate of  error rate: 7.46%
Confusion matrix:
    0   1   2   3   4   5   6   7   8   9 class.error
0 256   0   1   1   1   1   4   1   2   0  0.04119850
1   0 325   0   2   0   0   0   1   1   0  0.01215805
2   0   2 235   6   2   1   3   5   2   2  0.08914729
3   0   0   6 276   1   8   1   7   9   1  0.10679612
4   0   0   1   0 258   0   5   1   1   9  0.06181818
5   1   3   1  16   3 220   3   1   3   1  0.12698413
6   2   0   1   0   0   4 286   0   0   0  0.02389078
7   0   2   5   0   4   0   0 279   0  11  0.07308970
8   2   4   1   7   1   8   0   1 241   5  0.10740741
9   2   2   0   5   8   2   0   8   4 215  0.12601626
```

Ta thấy tỉ lệ lỗi nhiều nhất ở số 5, 8 và 9. Độ chính xác chúng ta nhận được là 0.925, tương đối cao chỉ với 10% tập dữ liệu.

```
[1] 0.9251032
```



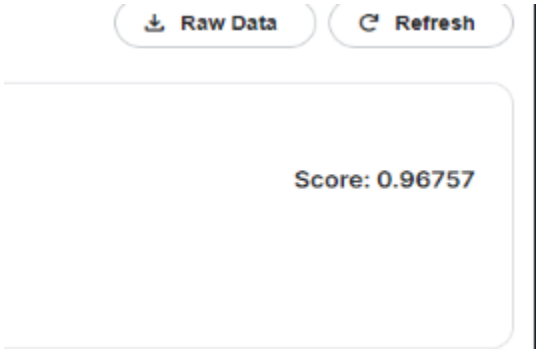
Dựa vào biểu đồ này chúng ta thấy được rằng số cây từ 0 đến 100 thì tỉ lệ lỗi khá cáo và không ổn định nhưng từ 100 cây trở lên thì tỉ lệ lỗi thấp dần đi và dường như không có nhiều biến động.

```
          0           1           2           3           4
0.01580264 0.01624835 0.04175341 0.06042250 0.03568517

          5           6           7           8           9
0.04829786 0.02235979 0.03985254 0.05779328 0.06132590
```

Tỉ lệ lỗi với từng nhãn, ta thấy nhãn 0 và 1 có tỷ lệ lỗi thấp nhất do đó đây là 2 nhãn dễ dự đoán nhất, còn nhãn khó dự đoán nhất là 8 và 9.

Sau khi submit trên cuộc thi ở Kaggle thì ta được kết quả:



Kết quả này tốt, mô hình Random Forest này rất tốt đối với bài toán này,

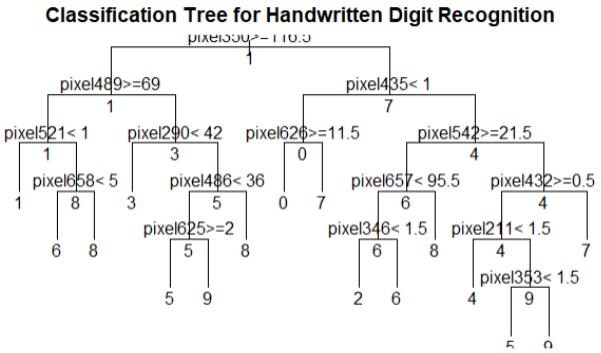
Thuật toán Decision Tree: ta cùng sử dụng 10% tập dữ liệu để đánh giá mức độ hiệu quả của thuật toán.

```
          Reference
Prediction   0   1   2   3   4   5   6   7   8   9
         0 107   0   9   4   0  14   2   5   2   2
         1   0 123   6   1   5   3   1   5  18   7
         2   0  10  67   1   0   3  23   8   2   2
         3   4   8   7  79   2  15   7   0  17   2
         4   0   1   2   0  70  10   5   1   1   1
         5   4   0   2  11   4  51   1   0   4   6
         6   1   2  14   1  11   4  93   7   8   7
         7   9   7   6   7  27  13  19  91   3  24
         8   2   5  15   5   5  12  10   5  71   4
         9   0   1  11  17  26   4   1   9  13  85
```

Dựa vào ma trận nhầm lẫn thì chúng ta thấy rằng các nhãn dự đoán sai khá nhiều và chúng ta nhận được độ chính xác hơn đoán mò khoảng 0.1.

```
          Accuracy : 0.5979
            95% CI : (0.5716, 0.6237)
No Information Rate : 0.1157
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.5531
```



Việc vẽ cây dựa vào các thuộc tính quan trọng này không thực sự hợp lý đối với tập dữ liệu này. Kết quả này không tốt và mô hình này sẽ không tốt đối với bài toán này.

6. CONCLUSION

Sau khi thực nghiệm với 3 mô hình Decision Tree, Random Forest, và Knn thì ta thấy được 2 mô hình Random forest và Knn cho ra kết quả tốt nhất đối với bài toán này. Hai thuật toán này sử dụng dụng rất tốt và hiệu quả, đạt được kết quả cáo trong cuộc thi trên Kaggle.

Tại sao Decision Tree lại chỉ cho độ chính xác chỉ 0.63? Vì tập dữ liệu tương đối nhiều các thuộc tính và có nhiều thuộc tính chỉ toàn giá trị 0 nên nó sẽ ảnh hưởng rất nhiều đối với độ chính xác của mô hình Decision Tree.

Nếu có thêm tài nguyên tính toán, nhóm sẽ dự định thêm một vài mô hình để tăng độ chính xác với bài toán này. Một số mô hình như Support Machine Vector,

Neural Network. Sử dụng hình ảnh để nhận biết thay vì tập dữ liệu là pixel của hình ảnh đó.

7. APPENDICES

Phụ lục 1: Thành phần tập dữ liệu

Description: df [4,200 × 785]

pixel0 <dbl>	pixel1 <dbl>	pixel2 <dbl>	pixel3 <dbl>
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	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

1-10 of 4,200 rows | 1-10 of 785 columns

Phụ lục 2: Mẫu đầu ra để submit lên cuộc thi trên Kaggle.

	A	B
1	ImageId	Label
2	1	2
3	2	0
4	3	9
5	4	9
6	5	3
7	6	7
8	7	0
9	8	3
10	9	0

Phụ lục 3: Tổng thời gian thực thi.

```
   user  system elapsed
3991.53    9.08 4011.36
```

Phụ lục 4: Các chỉ số thống kê bởi từng nhãn trong bài toán.

```
Statistics by Class:

                     Class: 0 Class: 1 Class: 2 (
Sensitivity           0.97638   0.9682  0.93525
Specificity           0.99372   0.9960  0.99128
Pos Pred Value        0.93939   0.9682  0.92199
Neg Pred Value        0.99763   0.9960  0.99285
Prevalence            0.09071   0.1121  0.09929
Detection Rate        0.08857   0.1086  0.09286
Detection Prevalence  0.09429   0.1121  0.10071
Balanced Accuracy     0.98505   0.9821  0.96326
```

Phụ lục 5: Độ quan trọng của các biến trong mô hình Decision Tree

```
   pixel413    pixel353    pixel386    pixel385    pixel441
148.7965182 147.0505051 140.9132590 136.9716293 134.0154071
   pixel358    pixel515    pixel542    pixel514    pixel543
128.1029627 110.4456088 106.5011228 100.5843937  92.6954217
   pixel541     pixel98     pixel99     pixel97    pixel124
 88.7509357  71.8168801  69.8492943  66.8979157  65.9141228
   pixel741    pixel122     pixel71     pixel72    pixel740
 63.1262921  61.9789513  59.3648542  59.3648542  59.1808988
   pixel101    pixel739     pixel70     pixel73    pixel742
 58.3754400  58.1945505  56.3966115  56.3966115  53.2628089
   pixel738    pixel174    pixel173    pixel677    pixel146
 51.2901123  15.7920209  13.8180183  10.8570144   9.8700131
   pixel648    pixel666    pixel312    pixel638
  9.8700131   1.9687974   0.9843987   0.9843987
```

CONTRIBUTIONS

MSSV	STT	NAME	MAIN	PROGRESS
19133010	04	Trần Nguyên Thái Bảo	Intro, Data Modeling(Random Forest)	100%
19133023	20	Lê Tuấn Hiệp	Data Modeling(Decision Tree)	100%
19133025	21	Đinh Quốc Hùng	Visualization Data Modeling(Knn)	100%

REFERENCES

[1] Introduction to Data Mining by Vipin Kumar, Pang-Ning Tan, Michael Steinback, Anuj Karpatne (z-lib.org)

[2] Data Mining Methods and Models 1st Edition by DANIEL T. LAROSE.

[3] Quach, Dinh Hoang. Lecture, 2022. Slide and video.

[4] Chang, Winston. *R Graphic Cookbook 2nd*. NHÀ XUẤT BẢN O'reilly, 2021. EBOOK from https://r-graphics.org/.

[5] Handwriting recognition from https://en.wikipedia.org/wiki/Handwriting_recognition.