ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

Môn: Học máy

Phân lớp ảnh chữ số viết tay bằng Support Vector Machine

Nhóm thực hiện (nhóm 1):

Bùi Chí Dũng – 1712364

Nguyễn Công Lý – 1712584

8 / 2020

I. QUÁ TRÌNH THỰC HIỆN

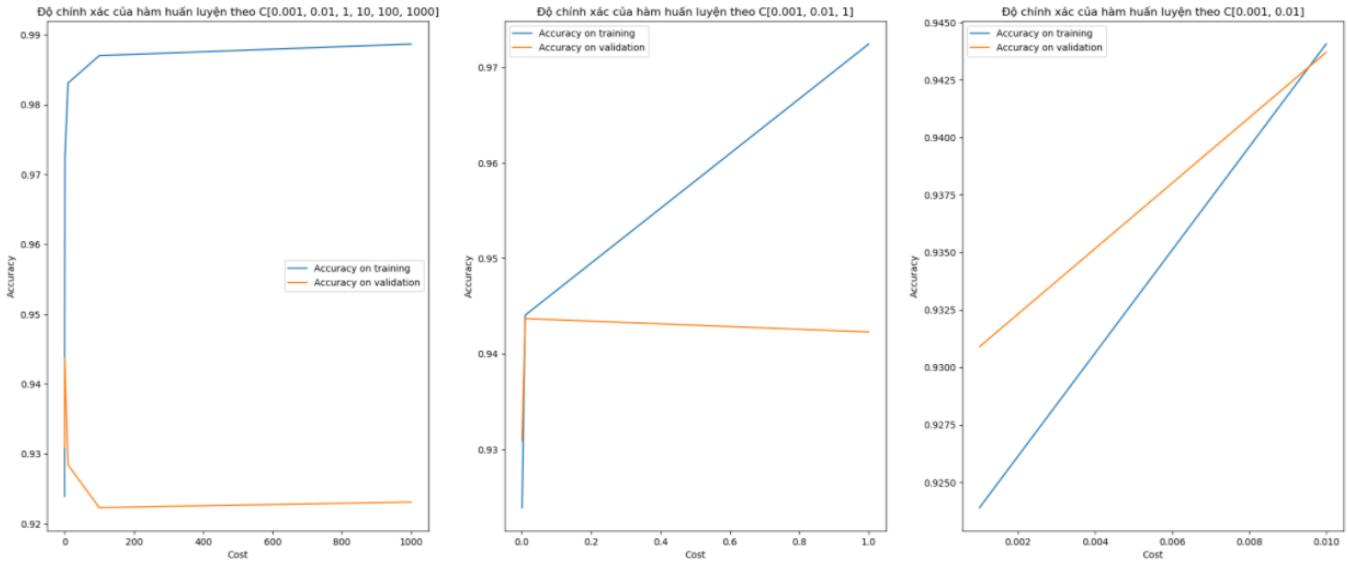
STT	Công việc	Ngày thực hiện	Người thực hiện
1	Tìm hiểu, ôn tập cơ sở lý thuyết SVM	4/8 – 15/8	Dũng, Lý
2	Thảo luận, phân chia công việc	16/8	Dũng, Lý
3	Linear Kernel, các thành viên đều train với cùng một khoảng C. So sánh kết quả, chỉnh sửa, tối ưu code.	20/8 – 24/8	Dũng, Lý
4	Cập nhật kế hoạch, sửa lỗi, tham khảo tài liệu.	22/8	Lý
5	Vẽ biểu đồ đánh giá độ chính xác theo sự thay đổi của siêu tham số C (Linear).	24/8	Dũng
6	RBF Kernel, thống nhất khoảng giá trị của C và Gamma để tiến hành train, các thành viên đều thực hiện train và đối sánh kết quả.	22/8 – 26/8	Dũng, Lý
7	Đối sánh chiếu kết quả, thống nhất kết quả và cập nhật vào báo cáo, vẽ biểu đồ sự thay đổi độ chính xác của RBF	25/8	Dũng
8	Hoàn thiện báo cáo	26/8	Lý
9	Hoàn thiện file yêu cầu và nộp đồ án	26/8	Dũng

II. HUẤN LUYỆN SVM

1. Dùng linear kernel (không dùng kernel)

Dưới đây là kết quả (độ lỗi trên tập training và tập validation) thu được sau khi train lần lượt dữ liệu bằng Linear Kernel trên tập training với các siêu tham số **C thay đổi từ 0.001, 0.01, 1, 10, 100, 1000**.

```
Với C =  0.001
Total time: 725.2277231216431 seconds
Độ lỗi trên tập training:  0.07609999999999995
Độ lỗi trên tập validation:  0.06910000000000005
Với C =  0.01
Total time: 359.16287207603455 seconds
Độ lỗi trên tập training:  0.05593999999999999
Độ lỗi trên tập validation:  0.05630000000000002
Với C =  1
Total time: 321.9312324523926 seconds
Độ lỗi trên tập training:  0.02754000000000001
Độ lỗi trên tập validation:  0.057699999999999974
Với C =  10
Total time: 432.980797290802 seconds
Độ lỗi trên tập training:  0.016920000000000046
Độ lỗi trên tập validation:  0.0716
Với C =  100
Total time: 1076.4254505634308 seconds
Độ lỗi trên tập training:  0.013000000000000012
Độ lỗi trên tập validation:  0.07769999999999999
Với C =  1000
Total time: 8670.351508378983 seconds
Độ lỗi trên tập training:  0.011340000000000017
Độ lỗi trên tập validation:  0.07689999999999997
```



Biểu đồ sự thay đổi độ chính xác theo các giá trị C

Từ bảng kết quả và biểu đồ, ta nhận thấy:

- Độ chính xác trên tập training tăng dần (hay nói cách khác là độ lỗi giảm dần) khi ta tăng giá trị siêu tham số C.

- Đối với độ chính xác trên tập validation, ta nhận thấy kết quả tốt nhất thu được với **C = 0.01**, sự thay đổi bắt đầu với **C = 0.001**, cho kết quả độ chính xác tăng dần cho đến **C = 0.01**, và sau đó lại giảm dần khi giá trị C tăng lên.

- Tuy nhiên, ta cũng thấy khi giá trị C tiến về C = 1000 và hơn nữa, độ chính xác trên cả tập training và tập test đều tăng lên

Đối với Support Vector Machine sử dụng Linear Kernel:

- C = inf: Không cho phép sai lệch, đồng nghĩa với **Hard Margin**, ưu tiên phân loại đúng

- C lớn: Cho phép sai lệch nhỏ, thu được Margin nhỏ.

- C nhỏ: Cho phép sai lệch lớn, thu được Margin lớn.

- C trung bình: Cho phép sai lệch vừa phải, Margin vừa phải.

Có nghĩa là theo lý thuyết thì khi giá trị C cao, có nghĩa là chúng ta ưu tiên phân loại đúng hơn là độ lớn của Margin hyperplane. Khi margin lớn, có nghĩa là những quan sát (hay các điểm dữ liệu) trong tương lai có thể nằm đâu trong khoảng margin hyperplane, và khi đó có thể vẫn được phân loại đúng. Còn khi chúng ta ưu tiên phân loại đúng ngay từ đầu trên tập training, tức là cho C nhỏ, thì khi kiểm tra trên tập test có thể dễ bị phân loại sai, do margin hyperplane nhỏ, và các điểm dữ liệu mới lại nằm gần ranh giới hơn so với các điểm đã training. Tuy nhiên cũng không có một quy luật nào về giá trị nào của C thì thu được model tốt. Mà phụ thuộc nhiều vào tập dữ liệu và độ lỗi trên tập dữ liệu đó.

Đối với thời gian chạy, theo như kết quả nhóm thu được, thời gian chạy có thể ảnh hưởng bởi giá trị tham số, tuy nhiên cũng một phần ảnh hưởng bởi các yếu tố khác như phần cứng máy tính, cho kết quả không đều giữa các giá trị C. Tuy nhiên với C càng lớn, thời gian chạy cũng tăng lên rất nhiều. Đối với nhóm đã thực hiện các giá trị C > 1000, cho vài kết quả lớn hơn 4 giờ, nên không đề cập trong phần đánh giá.

```
Với C =  1000
Total time: 8215.976546764374 seconds
Độ lỗi trên tập training:  0.011340000000000017
Độ lỗi trên tập validation:  0.07689999999999997
Với C =  1500
Total time: 12447.431423664093 seconds
Độ lỗi trên tập training:  0.011519999999999975
Độ lỗi trên tập validation:  0.07709999999999995
Với C =  2000
Total time: 16469.863746881485 seconds
Độ lỗi trên tập training:  0.011499999999999955
Độ lỗi trên tập validation:  0.07750000000000001
Với C =  2500
```

```
clf = svm.SVC(C = 0.00000001, kernel = 'linear') #Linear Kernel

start_time = time.time()
clf.fit(train_X, train_Y)
print("Total time: %s seconds" % (time.time() - start_time))

#Get the error value
E_train_c = 1 - clf.score(train_X, train_Y)
E_val_c = 1 - clf.score(val_X, val_Y)

print("Độ lỗi trên tập training: ", E_train_c)
print("Độ lỗi trên tập validation: ", E_val_c)
```

```
Total time: 4749.482308387756 seconds
Độ lỗi trên tập training:  0.88644
Độ lỗi trên tập validation:  0.8936
```

```
print("Độ lỗi trên tập testing: ", 1 - clf.score(test_X, test_Y))
```

```
Độ lỗi trên tập testing:  0.8865
```

Bên trên là thời gian chạy, cũng như độ lỗi với C nhỏ = 0.00000001 và C > 1000. Cho thấy với C nhỏ thì độ lỗi rất lớn, và C lớn thì độ lỗi thấp, tuy nhiên biến đổi rất chậm, thời gian chạy thì rất lâu, tuy nhiên nếu có thời gian nhóm sẽ thử chạy những giá trị C lớn hơn nữa để đối sánh kết quả với lý thuyết đã học.

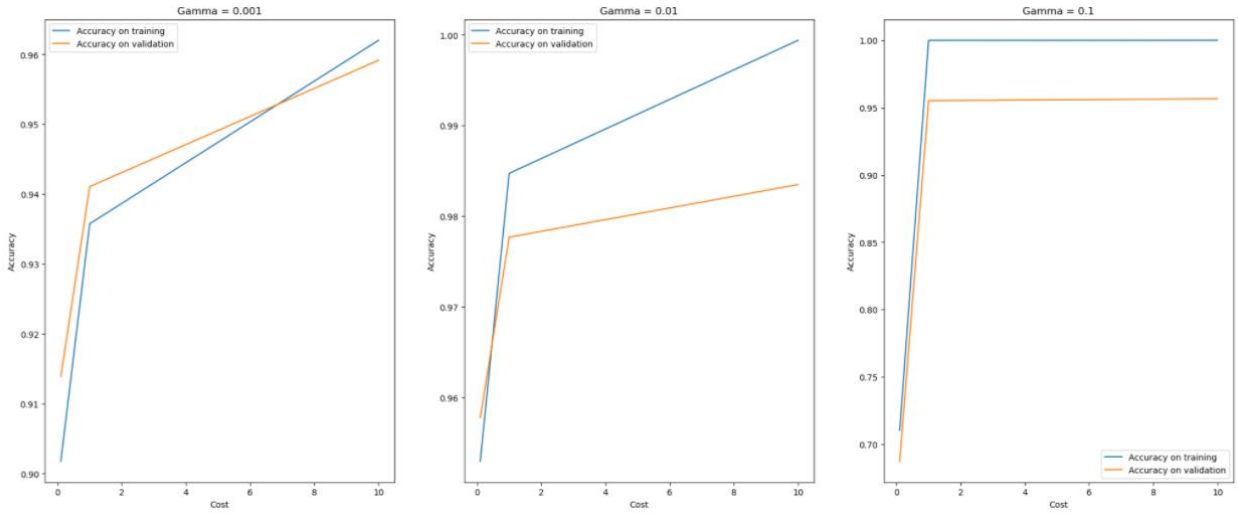
Từ kết quả trên, ta chọn được **c_best = 0.01** là tốt nhất (độ lỗi thấp nhất trong các giá trị C sử dụng) cho tập dữ liệu được thực hiện, lấy giá trị C này để dùng làm tham số trong hàm dự đoán cuối cùng.

2. Dùng Gaussian/RBF kernel

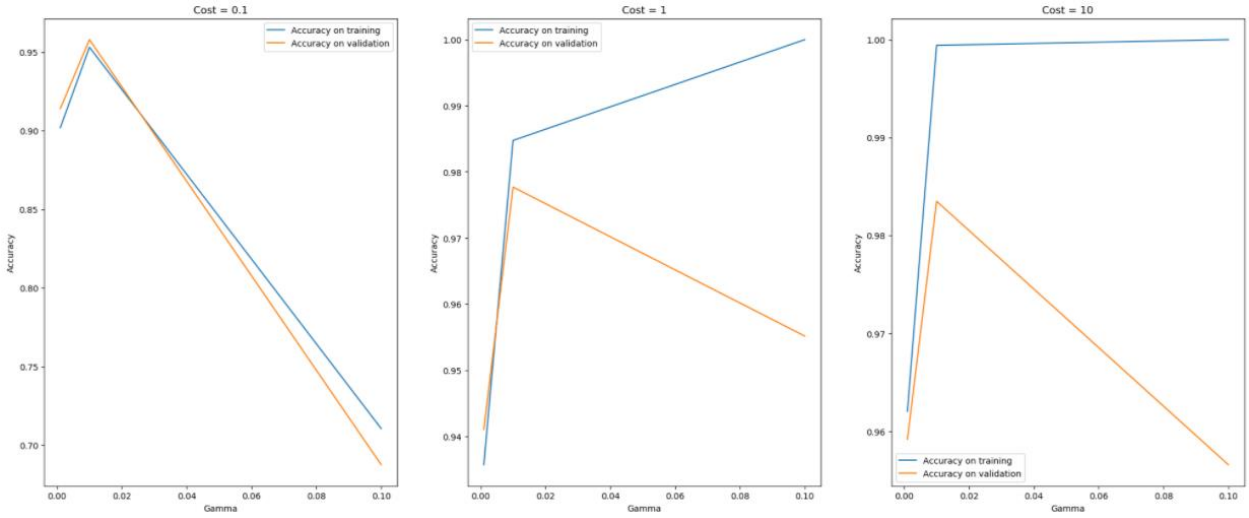
Dưới đây là kết quả thu được (độ lỗi trên tập training và tập validation) sau khi train dữ liệu bằng Gaussian/RBF Kernel trên tập training với các tham số: **C = [0.1, 1, 10] và gamma = [0.001, 0.01, 0.1].**

```
Với C =  0.1
Với gamma =  0.001
Total time: 1529.9412128925323 seconds
Độ lỗi trên tập training:  0.09824
Độ lỗi trên tập validation:  0.08609999999999995
Với C =  0.1
Với gamma =  0.01
Total time: 702.927503824234 seconds
Độ lỗi trên tập training:  0.04701999999999995
Độ lỗi trên tập validation:  0.042200000000000015
Với C =  0.1
Với gamma =  0.1
Total time: 3620.643173456192 seconds
Độ lỗi trên tập training:  0.28952
Độ lỗi trên tập validation:  0.3125
Với C =  1
Với gamma =  0.001
Total time: 551.3761706352234 seconds
Độ lỗi trên tập training:  0.06422000000000005
Độ lỗi trên tập validation:  0.05889999999999995
Với C =  1
Với gamma =  0.01
Total time: 314.2866096496582 seconds
Độ lỗi trên tập training:  0.015260000000000051
Độ lỗi trên tập validation:  0.022299999999999986
Với C =  1
Với gamma =  0.1
Total time: 5576.588743448257 seconds
Độ lỗi trên tập training:  4.0000000000040004e-05
Độ lỗi trên tập validation:  0.04479999999999995
Với C =  10
Với gamma =  0.001
Total time: 323.1787791252136 seconds
Độ lỗi trên tập training:  0.037939999999999974
Độ lỗi trên tập validation:  0.04079999999999995
Với C =  10
Với gamma =  0.01
Total time: 257.4736008644104 seconds
Độ lỗi trên tập training:  0.0005800000000000249
Độ lỗi trên tập validation:  0.01649999999999996
Với C =  10
Với gamma =  0.1
Total time: 5556.750864505768 seconds
Độ lỗi trên tập training:  0.0
Độ lỗi trên tập validation:  0.043399999999999994
```

Bảng tổng hợp kết quả độ lỗi trên tập training và validation



Biểu đồ đánh giá độ chính xác theo C



Biểu đồ đánh giá độ chính xác theo gamma

Theo cơ sở lý thuyết, đối với RBF kernel, giá trị siêu tham số C có ảnh hưởng tương tự đến model như đã đề cập bên trên phần Linear Kernel. Ở RBF kernel, xuất hiện thêm 1 siêu tham số đó là gamma, tham số này quyết định độ cong của ranh giới phân chia các class, với gamma lớn, tức là độ cong sẽ lớn (**high bias** – underfit và cho độ chính xác thấp khi dự đoán, low variance), gamma nhỏ thì ngược lại (low bias, **high variance** – overfit, độ chính xác thấp). Tuy nhiên không thể căn cứ vào gamma để tìm ra một hàm dự đoán tốt, mà việc đó phụ thuộc nhiều vào dữ liệu cùng các yếu tố khác.

Như vậy, đối với việc sử dụng Gassian/Kernel thì theo biểu đồ ta thấy, đối với sự thay đổi của siêu tham số C, khi **C tăng dần từ 0.1 đến 10**, độ chính xác đạt gần như cực đại trên tập training (độ lỗi bằng 0) tại **C = 10 ở gamma bằng 0.1,** ở các biểu đồ **gamma = 0.001 và 001**, cho giá trị chính xác tỉ lệ thuận với chiều hướng tăng của giá trị C. Độ chính xác trên tập validation trong trường hợp này cũng tỉ lệ thuận theo giá trị tăng của C.

Đối với sự thay đổi của gamma, ta thấy gần như độ chính xác đạt cực đại ở tại **gamma = 0.01** ở cả 3 đồ thị, tương ứng với 3 giá trị của **C là 0.1, 1 và 10** trên cả tập training và tập validation. Ở c = 10, độ chính xác trên tập training gần bằng 100% với gamma = 0.01 và độ chính xác trên tập validation cao nhất trong các trường hợp.

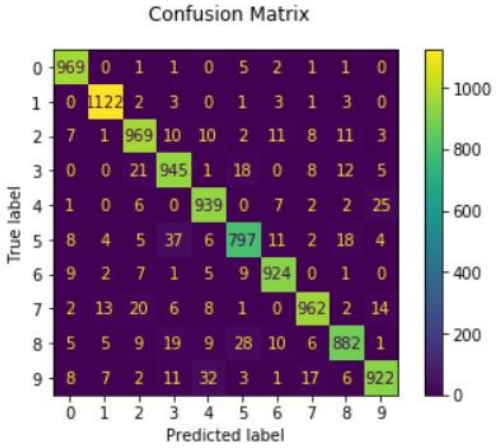
Đối với thời gian chạy, kết quả nhóm thu được cho thấy, ở các giá trị gamma = 0.1, có thời gian chạy rất lâu so với các giá trị khác của gamma, và cả giá trị khác của C. Đối với giá trị gamma thấp hơn, thời gian chạy cũng tương đối, tuy nhiên độ lỗi vẫn còn ở mức cao. Còn đối với gamma lớn hơn 0.1, thời gian chạy > 15000s nên nhóm không đề cập vào kết quả. Nếu có thời gian nhóm sẽ train thêm với nhiều giá trị của c và gamma để đối sánh kết quả.

Từ đó, ta lấy c_best = 10 và gamma_best = 10 là 2 tham số cho hàm huấn luyện cuối cùng.

III. ĐÁNH GIÁ SVM

1. Linear Kernel

Sau khi chọn được **c_best**, dùng c_best làm giá trị của C trong hàm huấn luyện, và ta thu được kết quả độ chính xác, cũng như độ lỗi trên tập test như sau:



Với C = 0.01

Total: 385.20823192596436 seconds

Độ lỗi trên tập test: 0.05689999999999995

Test error rate nằm ở mức: **0.05689995**, tức là khoảng **5.689%**

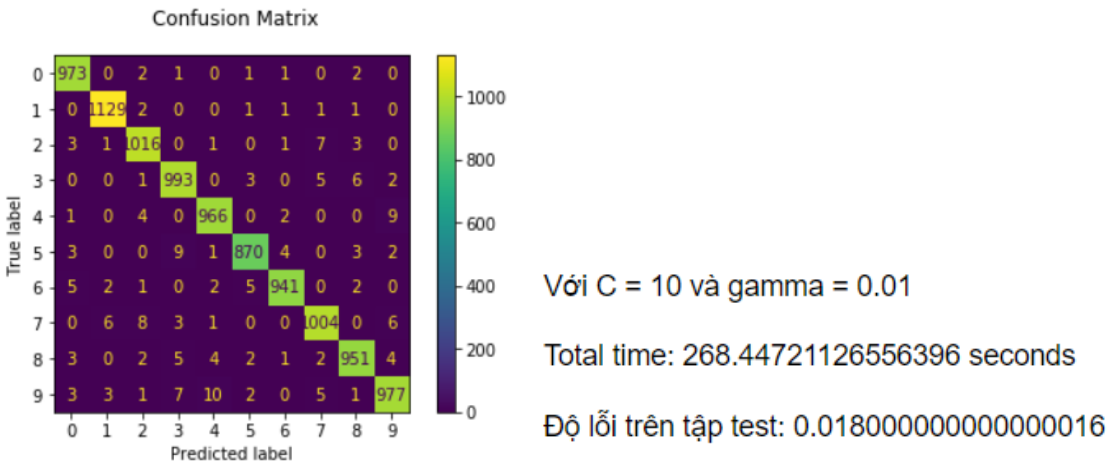
Kết quả trên cho thấy, trên tập test gồm 10000 mẫu ảnh viết tay, cho thấy (label là các chữ số từ 0 đến 9 theo bộ dữ liệu MNIST):

- Đối với dòng đầu tiên, label '0' (label đúng là '0'), số lượng dự đoán đúng là 969, và các dự đoán sai bao gồm: 1 ảnh là label '2', 1 ảnh là label '3', 5 ảnh là label '5', 2 ảnh là label '6', 1 ảnh là label '7' và 1 ảnh là label '8'.

- Các dòng tiếp theo, cho kết quả dự đoán đúng lần lượt là: 1122 dự đoán đúng ở label '1', 969 dự đoán đúng ở label '2', 945 dự đoán đúng ở label '3', 939 dự đoán đúng ở label '4', 797 dự đoán đúng ở label '5', 924 dự đoán đúng ở label '6', 962 dự đoán đúng ở label '7', 882 dự đoán đúng ở label '8' và 922 dự đoán đúng ở label '9'

2. Gaussian/RBF

Sau khi chọn được c_best = 10 và gamma_best = 10 là 2 tham số cho hàm huấn luyện cuối cùng. Ta tiến hành train dữ liệu trên tập traning với Gaussian/RBF Kernel sử dụng 2 tham số này và thu được kết quả như sau:



Test error rate nằm ở mức **0.018**, tương ứng với khoảng **1.8%**, một kết quả khá khả quan.

Kết quả từ biểu đồ trên cho thấy trên tập test gồm 10000 ảnh mẫu viết tay, cho thấy (label là các chữ số từ 0 đến 0 theo bộ dữ liệu MNIST):

- Đối với dòng đầu tiên, label '0' (label đúng là label '0'), với số lượng dự đoán đúng là 973, và các dự đoán sai bao gồm: 2 label '2', 1 label '3', 1 label '5', 1 label '6', 2 label '8'.

- Các dòng tiếp theo thể hiện các dự đoán đúng như: 1129 dự đoán đúng về label '1', 1016 label '2', 993 label '3', 966 label '4', 870 label '5', 941 label '6', 1004 label '7', 951 label '8' và 977 label '9'

Nhìn chung, kết quả thu được khi sử dụng **Gaussian/RBF Kernel** (Error: 5.689%) cho kết quả khả quan hơn rất nhiều so với dùng **Linear Kernel** (Error: 1.8%), tương ứng với đó là các mẫu ảnh bị phân loại sai ở RBF ít hơn khá nhiều so với Linear. Các siêu tham số ảnh hưởng đến

IV. TÀI LIỆU THAM KHẢO

- https://work.caltech.edu/telecourse (LEARNING FROM DATA - Machine Learning Course, Taugh by Caltech Professor Yaser Abu-Mostafa)

- https://scikit-learn.org/stable/ (Scikit-Learn)

- https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be (C and gamma in SVM)

- https://towardsdatascience.com/a-guide-to-svm-parameter-tuning-8bfe6b8a452c (C and gamma in SVM)

- https://allaravel.com/blog/ve-do-thi-trong-python-voi-thu-vien-matplotlib (Vẽ đồ thị trong python)

- https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (Confusion Matrix)

- https://github.com/lychengr3x/Digit-Classification-Using-SVM (by lychengr3x)

- https://github.com/akshayr89/MNSIST_Handwritten_Digit_Recognition-SVM (by akshayr89)

HẾT