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

CO3101

ĐỒ ÁN TỔNG HỢP - HƯỚNG TRÍ TUỆ NHÂN TẠO

Đề Tài
ỨNG DỤNG MACHINE LEARNING TRONG LỌC THƯ RÁC

GVHD: ThS. TRẦN NGỌC BẢO DUY

NHÓM THỰC HIỆN: Nhóm Bayesian network, L01

Nguyễn Duy Mạnh	1914121
Lê Đắc Thường	1915442
Lưu Trường Giang	1913186
Phan Vũ Hoàng Hiếu	1913355
Nguyễn Hữu Lợi	1914047

Tp. Hồ Chí Minh, Tháng 12/2021

Mục lục

1 Giới thiệu đề tài

1.1 Đôi nét sơ lược

Machine learning là một lĩnh vực con của Trí tuệ nhân tạo(Artificial Intelligence) sử dụng các thuật toán cho phép máy tính có thể học từ dữ liệu để thực hiện các công việc thay vì được lập trình một cách rõ ràng. Thời gian gần đây, Machine Learning ngày càng trở nên thông dụng và được ứng dụng ở rất nhiều lĩnh vực gần gũi với chúng ta, nó xuất hiện ở mọi nơi. Machine Learning đứng sau các trợ lý ảo, ứng dụng dịch ngôn ngữ, các chương trình mà YouTube gợi ý và nội dung hiển thị trên mạng xã hội của chúng ta. Nó có thể điều khiển các loại xe tự lái và chẩn đoán tình trạng bệnh lý dựa trên hình ảnh. Ở đề tài này, chúng ta sẽ ứng dụng Machine Learing vào lĩnh vực lọc thư rác.

Cùng với sự phát triển mạnh mẽ của mạng máy tính và thư điện tử, số lượng thư rác cũng tăng lên nhanh chóng, vượt quá sự phát triển của các công nghệ, dịch vụ chống thư rác. Do đó, việc nghiên cứu về bộ lọc thư rác tiên tiên, đem lại hiệu quả là điều thực sự cần thiết, đối với cả người dùng và nhà cung cấp dịch vụ. Với thực trạng như vậy, nhóm đã đặt vấn đề và nghiên cứu đề tài "Ứng dụng Machine Learning trong lọc thư rác" để vận dụng những giải thuật Machine Learning gây dựng những mô hình với khả năng lọc thư rác hiệu quả cao.

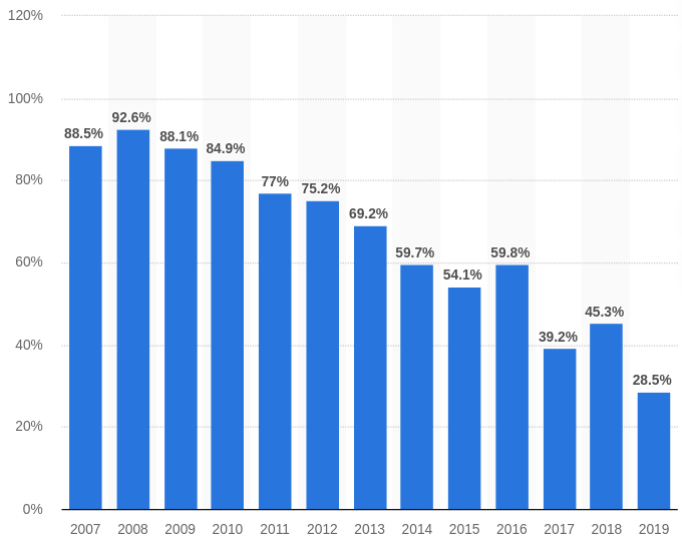
1.2 Vấn nạn thư rác

Email hay còn gọi là thư điện tử, đây là một hệ thống chuyển nhận thư từ qua các mạng máy tính. Thư điện tử có thể gửi được chữ, hình ảnh, âm thanh, nó có thể chuyển thông tin từ một máy nguồn tới một hay nhiều máy nhận thông qua mạng Internet.

Phát tán thư rác là hành vi gửi thư điện tử mà người nhận không mong muốn, thường với nội dung quảng cáo, được gửi hàng loạt với số lượng lớn tới một tập hợp người nhận không phân biệt. Kẻ phát tán thư rác có thể lấy địa chỉ thư điện tử từ các trang web, phòng chat, tập dữ liệu cá nhân bị rò rỉ,...

Thư rác gây ra nhiều phiền toái và thiệt hại, bao gồm, ngăn người dùng sử dụng tối ưu thời gian, dung lượng lưu trữ và băng thông mạng. Một số lượng lớn thư rác truyền trong mạng máy tính có thể phá hủy không gian nhớ của máy chủ thư điện tử, băng thông đường truyền, tài nguyên tính toán và thời gian sử dụng của thiết bị người dùng.

Theo thống kê từ Công ty dữ liệu và thị trường Statista (Đức - Công ty nổi tiếng về số liệu với khoảng 1.000.000 thống kê về hơn 80.000 chủ đề từ hơn 22.500 nguồn và 170 ngành khác nhau trên toàn thế giới), số lượng thư rác đã lên đến 92,6% trong tổng số lượng thư điện tử được gửi đi vào năm 2008, sau đó có xu hướng giảm dần trong những năm gần đây do sự phát triển của các hệ thống chống thư rác. Vào năm 2019, số liệu này đã giảm còn 28,5%.



Hình 1: *Thống kê tỷ lệ thư rác từ năm 2007 đến năm 2019 (Statista).*

Tuy nhiên, điểm yếu của bộ lọc thư rác là đôi khi thư hợp pháp có thể bị đánh dấu là thư rác, từ đó

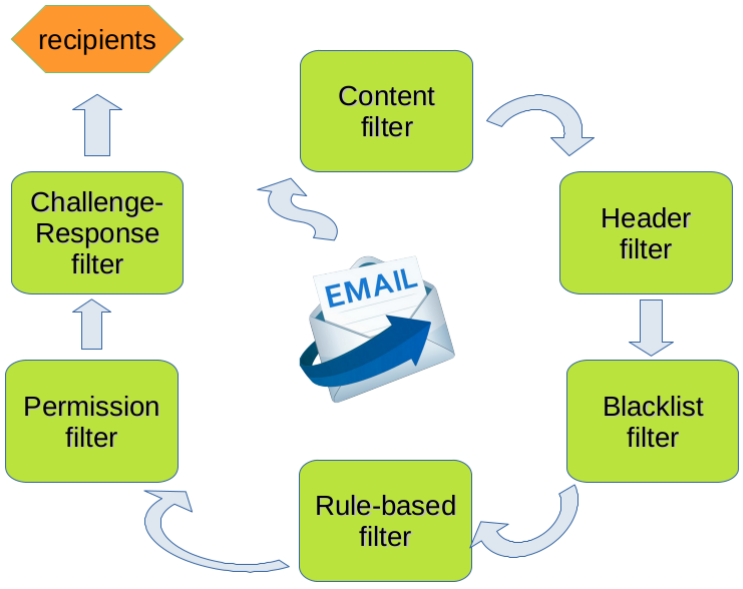
có thể bị từ chối hoặc loại bỏ. Chính vì vậy, mục đích của đề tài là nghiên cứu ứng dụng của học máy đối với lọc thư rác, bằng cách đưa ra một bức tranh tổng quan về thư rác, liệt kê những giải pháp đã có, chỉ ra nhu cầu của học máy trong việc lọc thư rác, trình bày việc áp dụng học máy đối với nhiệm vụ này thông qua một số thuật toán phổ biến như Naive Bayes, SVM và biến thể của SVM, ngoài ra còn sử dụng thêm Neural Network.

1.3 Cách thức hoạt động của bộ lọc thư rác

Bộ lọc thư rác hoạt động bằng cách phân tích thư trước khi được đưa vào hộp thư đến của người dùng nhằm kiểm tra xem có phải thư rác hay không. Bộ lọc này phân tích nội dung, địa chỉ gửi thư, đầu thư (header), các tệp đính kèm, ngôn ngữ và các dấu hiệu khả nghi khác.

Có nhiều phương pháp phân loại thư rác, trong đó quy trình lọc thư rác tiêu chuẩn bao gồm những thành phần sau:

- Lọc nội dung (content filter): phân loại thư rác dựa trên nội dung thư bằng các phương pháp như: thống kê truyền thống, học máy...;
- Lọc đầu thư (header filter): trích xuất thông tin từ đầu thư để lọc;
- Lọc theo danh sách đen (blacklist filter): xác định thư rác và địa chỉ thư gửi thư rác từ danh sách đen;
- Lọc dựa trên quy luật (rule-based filter): nhận diện người gửi cụ thể qua đầu thư sử dụng các tiêu chí do người dùng đặt ra;
- Lọc bằng cấp quyền (permission filter): gửi thư bằng cách có sự đồng ý của người nhận từ trước;
- Lọc nhờ xác thực bằng cách trả lời (challenge-response filter): kiểm tra thư được gửi tự động hay do người trực tiếp gửi (thông thường thư rác đều được tạo và gửi tự động thông qua phần mềm) qua việc người gửi trả lời để xác thực.



Hình 2: *Quy trình lọc thư rác tiêu chuẩn.*

Quy trình 6 thành phần nêu trên là quy trình tiêu chuẩn, tuy nhiên trong thực tế, những bộ lọc thư rác có thể không chỉ bao gồm các thành phần này.

Trong đó, một trong những phương pháp phổ biến từ phía người dùng được Gmail, Yahoo!, Outlook sử dụng là danh sách đen, chặn những địa chỉ hoặc tên miền mà người dùng cho là thư rác. Phương pháp khác cũng được sử dụng rộng rãi là lọc dựa trên quy luật. Trong Gmail, Yahoo! và Outlook, người

dùng có thể tạo danh sách trắng (whitelist) cho những người gửi đã biết trước. Ngoài ra, Outlook cho phép người dùng chặn những email có ngôn ngữ lạ thông qua Danh sách mã hóa đã bị chặn (Blocked Encodings List).

Mặc dù các bộ lọc thư rác ngày càng phát triển, nhưng người dùng vẫn có thể bị ngập trong thư rác mỗi ngày, vì kẻ phát tán thư rác có thể nhanh chóng thích ứng các kỹ thuật mới, cũng như do sự thiếu linh hoạt của các bộ lọc thư rác để thích nghi với sự thay đổi.

1.4 Mục tiêu của đề tài

Nội dung chính của đề tài này là xây dựng mô hình lọc thư rác tự động, dựa trên cơ sở sử dụng các mô hình Machine Learning (sẽ được trình bày ở dưới) để tính toán khả năng một thư điện tử có phải thư rác hay không. Khi có thư mới đến dữ liệu sẽ được tiền xử lý và làm sạch dữ liệu trước khi được đưa vào mô hình để phân loại.

Bài toán phân loại thư rác thực chất là bài toán phân loại văn bản hai lớp, trong đó tập cơ sở dữ liệu ban đầu là các thư rác và thư hợp lệ. Ý tưởng của phương pháp là tìm cách xây dựng một bộ phân loại nhằm phân loại cho một mẫu mới bằng cách huấn luyện từ những mẫu có sẵn. Ở đây mỗi mẫu ta xét đến chính là mỗi một email, tập các lớp mà mỗi email có thể là thư rác hoặc thư hợp lệ.

1.5 Tập dữ liệu

Bộ dữ liệu Ling-Spam là tập hợp của 2.893 thư, bao gồm thư rác và không phải thư rác được thu thập và sắp xếp từ các nhà ngôn ngữ học. Những email tập trung xung quanh các tin tuyển dụng, đơn xin học bổng nghiên cứu và thảo luận về phần mềm. Ngôn ngữ sử dụng là tiếng anh.
Bao gồm:

- 2412 thư hợp pháp
- 481 thư rác

Mỗi thư bao gồm phần tiêu đề (subject), nội dung (message) và nhãn (label). Nhãn ở đây là 0 nếu không phải là thư rác và 1 nếu thư đó là thư rác. Thông tin thêm về bộ dữ liệu, vui lòng truy cập ở đây Ling Spam dataset

2 Cơ sở lý thuyết

2.1 Naive bayes

Naive Bayes là một phương pháp đơn giản trong việc thiết lập một classifier. Phương pháp này sử dụng định lý Bayes:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Một cách trừu tượng, giả sử ta cần phân loại $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ trong đó $x_1, x_2, \cdots, x_n$ là các feature, kỹ thuật Naive Bayes sẽ tính

$$P(C|x_1, x_2, \cdots, x_n)$$

Trong đó C là một lớp nào đó, và sẽ phân loại vào lớp C cho ra giá trị trên là lớn nhất. Sử dụng định lý Bayes, ta có thể viết lại

$$P(C|x_1, x_2, \cdots, x_n) = \frac{P(x_1, x_2, \cdots, x_n|C)P(C)}{P(x_1, x_2, \cdots, x_n)}$$

Phương pháp Naive Bayes giả thiết rằng giá trị của một feature không phụ thuộc vào các feature còn lại nên ta có thể viết lại:

$$P(C|x_1, x_2, \cdots, x_n) = \frac{P(C) \cdot \prod P(x_i|C)}{P(x_1, x_2, \cdots, x_n)}$$

Ta thấu rằng mẫu số $P(x_1, x_2, \cdots, x_n)$ là như nhau với mọi lớp C, do đó ta chỉ cần so sánh giá trị $P(C) \cdot \prod P(x_i|C)$ của từng lớp C là chọn lớp C đưa ra giá trị lớn nhất để phân loại.

Những điểm mạnh của phương pháp này có thể kể đến như không yêu cầu tập dữ liệu quá lớn để hoạt động tốt, thời gian tính toán của Naive Bayes rất nhanh nên rất tốt cho dự đoán real-time, và rất dễ triển khai. Tuy nhiên Naive bayes cũng có những nhược điểm, chẳng hạn như việc giả sử các feature là độc lập với nhau, trong thực tế thì chuyện này gần như bất khả thi. Ngoài ra một vấn đề đó là nếu như có một x_i nào đó không xuất hiện ở phân lớp C trong tập dữ liệu, khi đó $P(x_i|C) = 0$ và làm cho $P(C|x_1, x_2, \cdots, x_n) = 0$ cho dù các feature khác cho ra $P(x_j|C)$ có cao đến mức nào. Để giải quyết vấn đề này, ta sẽ thường sử dụng kỹ thuật Laplace estimator:

$$P(x_i|c) = \frac{n_i + \alpha}{\sum(n_i + \alpha)} = \frac{n_i + \alpha}{n + \alpha \cdot d}$$

Với $\alpha > 0$. Khi đó nếu như $n_i = 0$ thì $P(x_i|c) = \frac{\alpha}{n + \alpha \cdot d} \neq 0$

Trong đó:

- n_i là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.
- n là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c.

2.1.1 Multinomial Naive Bayes

Với dữ liệu là dạng văn bản, ta phải biến đổi dữ liệu dạng này về dạng vector để mô hình học máy của chúng ta có thể sử dụng được. Ký hiệu $\mathbf{d}$ là một văn bản và $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N$ là các từ vựng. Giả thiết các từ vựng độc lập với nhau và thứ tự các từ vựng không ảnh hưởng, ta có

$$P(\mathbf{d}|C) = \frac{\left(\sum_{x=1}^{N} f_{x,\mathbf{d}}\right)!}{\prod_{x=1}^{N} f_{x,\mathbf{d}}!} \cdot \prod_{i=1}^{N} P(\mathbf{w}_i|C)^{f_{i,\mathbf{d}}}$$

Trong đó $f_{x,\mathbf{d}}$ là số lần suất hiện của $\mathbf{w}_x$ trong $\mathbf{d}$, $f_{i,\mathbf{d}}$ là số lần suất hiện của $\mathbf{w}_i$ trong $\mathbf{d}$.

2.1.2 Gaussian Naive Bayes

- Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.
- Với mỗi chiều dữ liệu i và một class c, x_i tuân theo một phân phối chuẩn có kỳ vọng μ_{ci} và phương sai σ_{ci}^2.

$$P(x_i|c) = p(x_i|\mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}}.exp(-\frac{(x_i - \mu_{ci}{}^2}{2\mu_{ci}^2})$$

- Trong đó, bộ tham số $\theta = \{\mu_{ci}, \sigma_{ci}^2\}$ được xác định bằng Maximum Likelihood.

$$(\mu_{ci}, \sigma_{ci}^2) = \arg\max_{\mu_{ci}, \sigma_{ci}^2} \prod_{n=1}^{N} P(\mathbf{x}_i^{(n)}|\mu_{ci}, \sigma_{ci}^2)$$

2.1.3 Bernoulli Naive Bayes

- Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary - bằng 0 hoặc 1. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không.
- Khi đó, $p(x_i|c)$ được tính bằng:

$$p(x_i|c) = p(i|c)^{x_i}(1 - p(i|c))^{1-x_i}$$

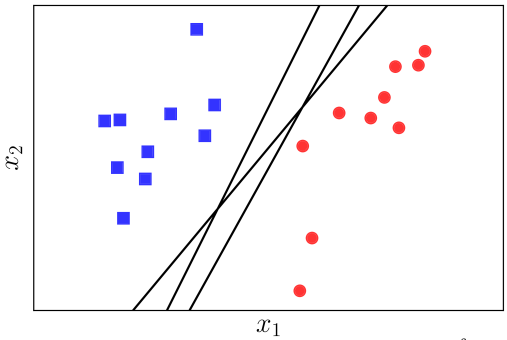
  với $p(i|c)$ có thể được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c.

2.1.4 Tóm tắt:

- Naive Bayes Classifiers (NBC) thường được sử dụng trong các bài toán Text Classification.
- NBC có thời gian training và test rất nhanh. Điều này có được là do giả sử về tính độc lập giữa các thành phần, nếu biết class.
- Nếu giả sử về tính độc lập được thoả mãn (dựa vào bản chất của dữ liệu), NBC được cho là cho kết quả tốt hơn so với SVM và logistic regression khi có ít dữ liệu training.
- NBC có thể hoạt động với các feature vector mà một phần là liên tục (sử dụng Gaussian Naive Bayes), phần còn lại ở dạng rời rạc (sử dụng Multinomial hoặc Bernoulli).
- Khi sử dụng Multinomial Naive Bayes, Laplace smoothing thường được sử dụng để tránh trường hợp một thành phần trong test data chưa xuất hiện ở training data.

2.2 Support vector machine

Ta nhắc lại bài toán đơn giản là phân chia lớp dữ liệu trên mặt phẳng 2 chiều:



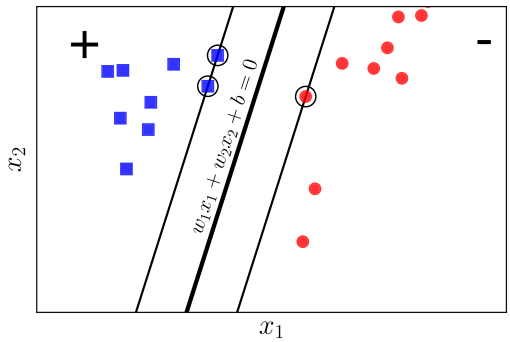
Hình 3: *Phân chia hai lớp dữ liệu trên mặt phẳng 2 chiều*

Ta thấy có rất nhiều cách phân chia. Nếu ta xét đến trường hợp chịu lỗi (tức có điểm phân loại sai thì càng có thêm nhiều đường thẳng phân chia 2 class). Vậy đâu là cách chia tốt nhất?
Thật khó để trả lời câu hỏi này, tuy nhiên ta vẫn có cách chia đủ tốt và đủ đơn giản đó là tạo ra một con đường với khoảng cách giữa 2 lề đường là margin không đổi. Nhận ra rằng nếu margin này càng lớn thì bộ phân loại của ta càng tốt. Hay nói cách khác là ta sẽ chọn một thẳng ở chính giữa làm phân chia và tối ưu hóa độ rộng 2 bên lề. Vì vậy ta dẫn đến một thuật toán gọi là Suport Vector Machine.

2.2.1 Mô hình hóa bài toán tối ưu cho SVM

Để dễ hình dung ở mặt phẳng d chiều thì ta xét nó trong 2 chiều rồi tổng quát hóa lên. Gọi các điểm trên mặt phẳng đó là $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ với class của nó là $y_i = 1$ hay $y_i = -1$

Dễ thấy ta luôn biểu diễn một đường thẳng ở dạng: $\mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$ là đường thẳng phân chia 2 class, chú ý ở đây ta có 2 hệ số cần đi tìm là w và b.



Hình 4: *Mô hình hóa bài toán ở góc độ hình học*

Ở góc độ hình học, giả sử các điểm màu xanh thuộc class 1, và điểm tròn thuộc class -1. Khi đó khoảng cách từ một điểm tới đường thẳng trung tâm phân chia đó là:

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

Muốn tạo ra một margin rộng nhất có thể thì ta xét tất cả các điểm xung quanh sao cho điểm gần nhất có khoảng cách đến đường đó lớn nhất. Khi đó thì ta xây dựng được công thức tối ưu:

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

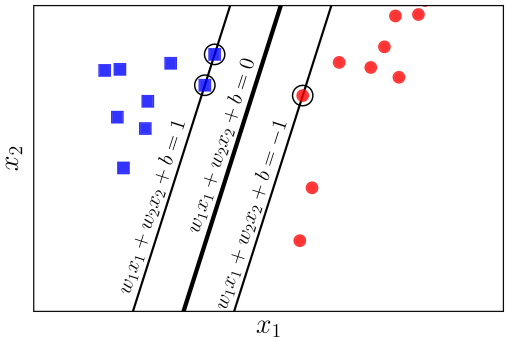
Phân tích tiếp ta có:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\} \quad (1)$$

Để ý thấy nếu w thay bằng kw và b thay bằng kb với k là hằng số dương thì mặt phẳng phân chia không hề thay đổi, tức là margin không đổi. Do đó nên ta có thể đơn giản hóa bài toán tối ưu bằng cách giả sử:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

Khi đó ta có như hình dưới đây:



Hình 5: *Mô hình hóa bài toán sau khi có những giả thiết*

Như vậy để ý thấy, những điểm nằm trên lề margin có khoảng cách tới đường phân chia trung tâm là 1, còn những điểm khác thì có khoảng cách tới đường phân chia trung tâm lớn hơn 1, do đó ta có:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$

Vậy bài toán tối ưu (1) có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||_2} \quad (1)$$

$$\text{subject to:} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \forall n = 1, 2, \ldots, N \quad (2) \quad (2)$$

Biến đổi tiếp ta được, để dễ đạo hàm ta nhân vế phải cho $\frac{1}{2}$

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2 \quad (3)$$

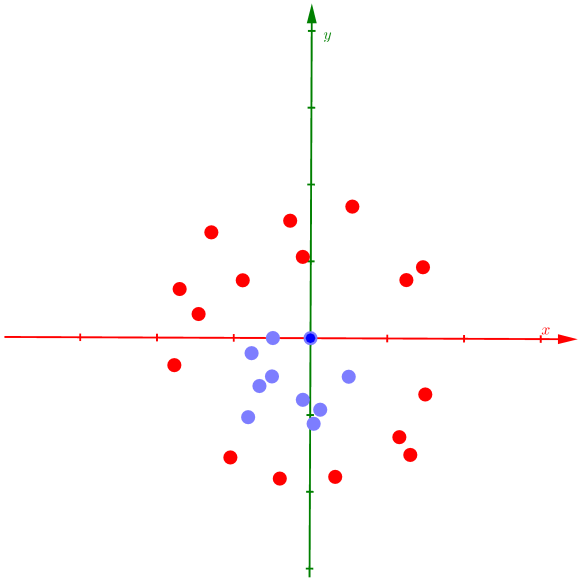
$$\text{subject to:} \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N \quad (3) \quad (4)$$

Có một vài phương pháp được đề xuất và ở đây điển hình là dùng bài toán đối ngẫu Lagrange kết hợp với điều kiện KKT, ở trong bài báo cáo này nhóm em xin được phép không trình bày mà tập trung vào khía cạnh ứng dụng thuật toán.

2.2.2 Phương pháp dùng Kernel Trick

Liệu rằng có phải tất cả mọi lúc thì ta có thể phân tách được 2 hay nhiều lớp dữ liệu bằng các đường tuyến tính hay không?

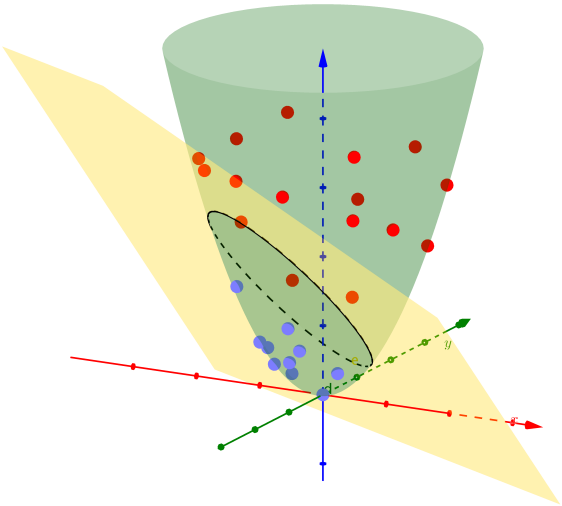
Câu trả lời là không! Ví dụ như:



Hình 6: *Hình ảnh về 2 lớp không thể dùng đường thẳng tuyến tính để phân loại*

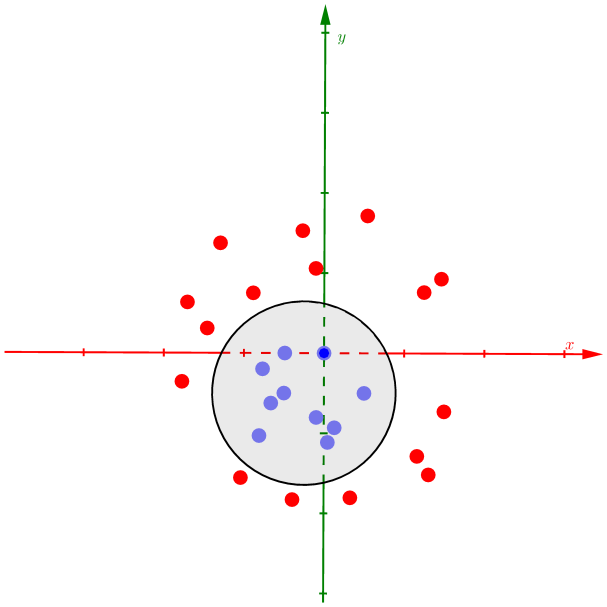
Ta thấy không thể tìm được đường thẳng tuyến tính nào có thể phân loại 2 lớp này cả, đến đây thì sẽ có một số phương pháp như dùng Soft Margin Support Vector Machine tức là ta sẽ chịu sẽ có lỗi trong việc phân loại, tức sẽ có một số điểm phân loại sai. Tuy nhiên nếu xét vào ví dụ hiện tại thì về mặt nhìn nhận ta thấy nếu có một đường thẳng nào phân biệt được hai class đó thì nó cũng không thể hiện được rằng là đó là một phép phân loại hợp lý. Bởi vì khi nhìn nhận ta thấy, cần một phép phân loại nào đó "không phải là đường thẳng" thì hợp lý hơn. Ở đây em đề xuất phương pháp dùng kernel trick để giải quyết bài toán này.

Kernel trick đơn giản là dùng một phép biến đổi nào đó để biến nó lên những chiều không gian khác sao cho ta tìm được một siêu phẳng để phân biệt rõ ràng 2 lớp dữ liệu đó.



Hình 7: *Chiếu tất cả điểm dữ liệu lên chiều không gian thứ 3*

Ta chọn chiều không gian thứ 3 với tính giá trị của nó bằng phép biến đổi: $z = x^2 + y^2$. Ta thấy tìm ra một mặt phẳng để phân chia được 2 lớp dữ liệu một cách rõ ràng hơn. Nếu ta chiếu xuống không gian 2 chiều ban đầu thì được một bộ phân loại "phi tuyến tính" trên không gian 2 chiều.



Hình 8: *Bộ phân loại "phi tuyến tính" trên không gian 2 chiều*

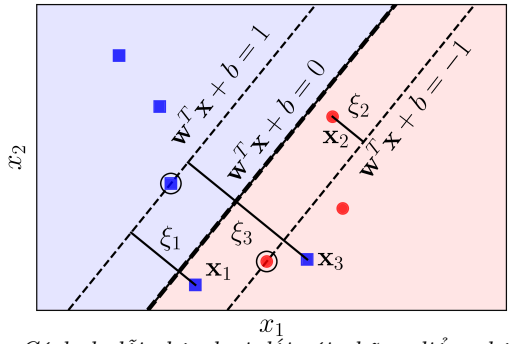
Khi tổng quát hóa lên không gian vector, ta thấy có 4 loại kernel trick thường gặp:

- Linear: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$
- Polynomial: $k(\mathbf{x}, \mathbf{z}) = (r + \gamma\mathbf{x}^T\mathbf{z})^d$
- Radial Basic Function: $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||_2^2), \quad \gamma > 0$
- Sigmoid: $k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma\mathbf{x}^T\mathbf{z} + r)$

Trong đó thì loại thứ 3 Radial Basic Function (viết tắt là rbf) là loại được ứng dụng rộng rãi và cho nhiều kết quả đa phần là tốt hơn những loại còn lại.

2.2.3 Soft Margin Support Vector Machine

Như đã có nói qua ở phần trước việc phân loại có chịu lỗi ở đây là rất thường xuyên xảy ra, với nhiều điểm dữ liệu, việc dùng kernel trick không thể đảm bảo chắc chắn rằng ta luôn phân chia được chúng thành các lớp khác nhau.



Hình 9: *Cách đo lỗi phân loại đối với những điểm phân loại sai*

Nhìn vào hình vẽ trên ta thấy những điểm mà ta chịu lỗi thì nó nhận một giá trị lỗi là:

$$\xi_i = |\mathbf{w}^T\mathbf{x}_i + b - y_i|$$

Khi đó hàm mục tiêu trở thành:

$$\frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{n=1}^{N}\xi_n$$

Với hằng số C>0 là hằng số chịu lỗi và $\xi = [\xi_1, \xi_2, \ldots, \xi_N]$. Khi đó ta được bài toán tối ưu với điều kiện:

$$(\mathbf{w}, b, \xi) = \arg\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{n=1}^{N}\xi_n \quad (5)$$

$$\text{subject to:} \quad 1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N \quad (2) \quad (6)$$

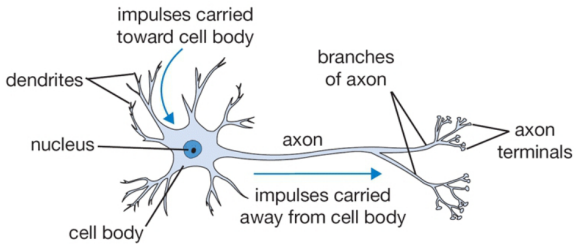
$$-\xi_n \leq 0, \ \forall n = 1, 2, \ldots, N \quad (7)$$

Ta thấy việc chọn lựa C càng nhỏ thì sự hy sinh của các điểm phân loại sai nó không ảnh hưởng nhiều, hay nói cách khác nó sẽ tập trung vào cái phần tổng khoảng cách đến đường phân chia sao cho nhỏ nhất có thể. Ngược lại với C lớn thì một sự hy sinh để chịu lỗi thì gây ảnh hưởng lớn đến kết quả, vì vậy khi tối ưu nó sẽ tập trung giảm thiểu phần lỗi phân loại.

2.3 Neural Network

2.3.1 Perceptron cơ bản

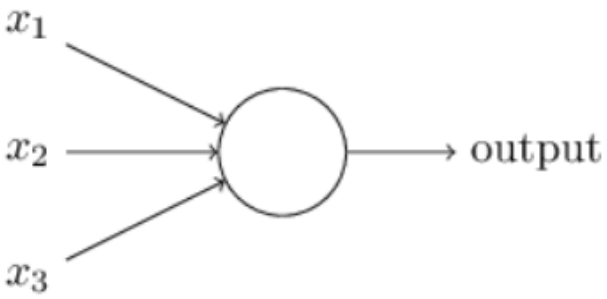
Một mạng Neural được cấu thành bởi các Neural đơn lẻ được gọi là các perceptron. Nên trước tiên ta tìm hiểu xem perceptron là gì đã rồi tiến tới mô hình của mạng Neural sau. Neural nhân tạo được lấy cảm hứng từ Neural sinh học như hình sau:



Hình 10: *mạng neural trong sinh học*

Quan sát hình ảnh dưới đây, ta có thể thấy một Neural có thể nhận nhiều đầu vào và cho ra một kết quả duy nhất. Mô hình của perceptron cũng tương tự như vậy.

Một perceptron sẽ nhận một hoặc nhiều đầu x vào dạng nhị phân và cho ra một kết quả o dạng nhị phân duy nhất. Các đầu vào được điều phối tầm ảnh hưởng bởi các tham số trọng lượng tương ứng w của nó, còn kết quả đầu ra được quyết định dựa vào một ngưỡng quyết định b nào đó.



Hình 11: *perceptron*

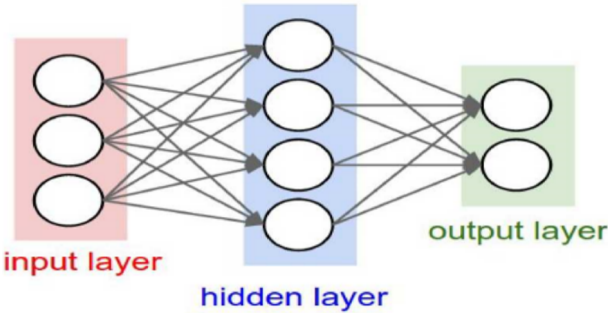
$$o = \begin{cases} 0 & \text{if } \sum_i w_i x_i \leq \text{threshold} \\ 1 & \text{if } \sum_i w_i x_i > \text{threshold} \end{cases}$$

Đặt $b = -\text{threshold}$, ta có thể viết lại thành:

$$o = \begin{cases} 0 & \text{if } \sum_i w_i x_i + b \leq 0 \\ 1 & \text{if } \sum_i w_i x_i + b > 0 \end{cases}$$

2.3.2 Kiến trúc mạng Neural nhân tạo

Mạng Neural là sự kết hợp của của các tầng perceptron hay còn được gọi là perceptron đa tầng (multilayer perceptron) như hình vẽ bên dưới.

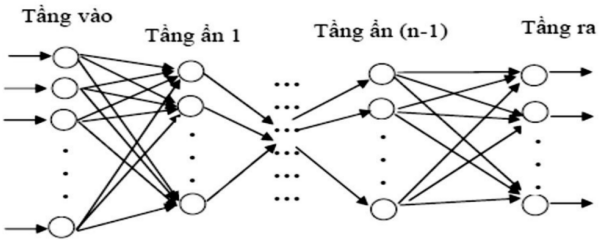


Hình 12: *multilayer perceptron*

Một mạng Neural sẽ có 3 kiểu tầng:

- Tầng vào (input layer): Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng;
- Tầng ra (output layer): Là tầng bên phải cùng của mạng thể hiện cho các đầu ra của mạng;
- Tầng ẩn (hidden layer): Là tầng nằm giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng.

Lưu ý rằng, một Neural chỉ có 1 tầng vào và 1 tầng ra nhưng có thể có nhiều tầng ẩn.



Trong mạng Neural, mỗi nút mạng là một sigmoid Neural nhưng hàm kích hoạt của chúng có thể khác nhau. Tuy nhiên trong thực tế người ta thường để chúng cùng dạng với nhau để tính toán cho thuận lợi.

Ở mỗi tầng, số lượng các nút mạng (Neural) có thể khác nhau tuỳ thuộc vào bài toán và cách giải quyết. Nhưng thường khi làm việc người ta để các tầng ẩn có số lượng Neural bằng nhau. Ngoài ra, các Neural ở các tầng thường được liên kết đôi một với nhau tạo thành mạng kết nối đầy đủ (full- connected network).

2.3.3 Kiến trúc mạng Neural MLP (Multi-layer Perceptron)

Mô hình mạng nơron được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng truyền thẳng (MLP: Multi Layer Perceptron). Một mạng MLP tổng quát là mạng có n (n>=2) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.

Kiến trúc của một mạng MLP tổng quát có thể mô tả như sau:

- Đầu vào là các vector (x1, x2, ..., xp) trong không gian p chiều, đầu ra là các vector (y1, y2, ..., yq) trong không gian q chiều. Đối với các bài toán phân loại, p chính là kích thước của mẫu đầu vào, q chính là số lớp cần phân loại;
- Mỗi neural thuộc tầng sau liên kết với tất cả các nơron thuộc tầng liền trước nó;
- Đầu ra của neural tầng trước là đầu vào của nơron thuộc tầng liền sau nó.

Hoạt động của mạng MLP như sau: tại tầng đầu vào các neural nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các neural thuộc tầng ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến tầng ẩn thứ 2. Quá trình tiếp tục cho đến khi các neural thuộc tầng ra cho kết quả.

2.3.4 Huấn luyện mạng MLP

2.3.4.a Khái niệm

Học là quá trình thay đổi hành vi của các vật theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai.

Một mạng neural được huấn luyện sao cho với một tập các vector đầu vào X, mạng có khả năng tạo ra tập các vector đầu ra mong muốn Y của nó. Tập X được sử dụng cho huấn luyện mạng được gọi là tập huấn luyện (training set). Các phần tử x thuộc X được gọi là các mẫu huấn luyện (training example). Quá trình huấn luyện bản chất là sự thay đổi các trọng số liên kết của mạng. Trong quá trình này, các trọng số của mạng sẽ hội tụ dần tới các giá trị sao cho với mỗi vector đầu vào x từ tập huấn luyện, mạng sẽ cho ra vector đầu ra y như mong muốn.

Có ba phương pháp học phổ biến là học có giám sát (supervised learning), học không giám sát (unsupervised learning) và học tăng cường (Reinforcement learning).

2.3.4.b Học có giám sát

Là quá trình học có sự tham gia giám sát của một "thầy giáo". Cũng giống như việc ta dạy một em nhỏ các chữ cái. Ta đưa ra một chữ "a" và bảo với em đó rằng đây là chữ "a". Việc này được thực hiện trên tất cả các mẫu chữ cái. Sau đó khi kiểm tra ta sẽ đưa ra một chữ cái bất kì (có thể viết hơi khác đi) và hỏi em đó đây là chữ gì?

Như vậy với học có giám sát, số lớp cần phân loại đã được biết trước. Nhiệm vụ của thuật toán là phải xác định được một cách thức phân lớp sao cho với mỗi vector đầu vào sẽ được phân loại chính xác vào lớp của nó.

3 Hiện thực và kết quả

3.1 Tiền xử lý email

Email thường có một số thông tin không hữu ích cho việc phân loại email. Để tăng độ chính xác cho việc phân loại, cần phải loại bỏ bớt các thành phần gây nhiễu, không hữu ích trong email. Do đó, ta sẽ thực hiện các bước sau để làm sạch email trước khi đưa vào huấn luyện.

Import một số thư viện cần thiết

```
import os, re, nltk
nltk.download('stopwords')
from nltk import corpus
from numpy.lib.function_base import vectorize
import pandas as pd
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
```

Đọc dữ liệu

Class Email có hai thuộc tính. Thuộc tính content lưu tiêu đề và nội dung của email sau khi ghép lại với nhau và thuộc tính label sẽ có giá trị là "1" khi email đó là email rác ngược lại có giá trị là "0". Khi một đối tượng Email được tạo ra sẽ thực hiện luôn quá trình tiền xử lý.

```
class Email:
    def __init__(self, content, label):
        self.content = content
        self.label = label
        self.preprocessing()
```

Hàm read() đọc dữ liệu trong tập tin messages.csv sau đó trả về danh sách các đối tượng của class Email với tiêu đề và nội dung email đã được ghép lại với nhau. Các dữ liệu bị khuyết sẽ bị loại bỏ bằng hàm dropna().

Hàm readTrainTestSplit() thực hiện việc đọc dữ liệu sau đó chuyển dữ liệu từ dạng ký tự thành vector bằng cách dùng CountVectorizer. CountVectorizer sẽ tạo ra một ma trận, mỗi cột tương ứng với một từ duy nhất, mỗi hàng tương ứng với một mẫu dữ liệu, giá trị ở mỗi ô là số lần xuất hiện của từ trong mẫu dữ liệu. Sử dụng ma trận này để đưa vào học máy sẽ dễ dàng hơn so với dạng ký tự. Sau khi chuyển thành dạng ma trận dùng hàm train_test_split() phân chia dữ liệu thành hai tập: một tập để huấn luyện và một tập để đánh giá lại kết quả huấn luyện.

```
@staticmethod
def read():
    if not os.path.exists('./data/messages.csv'):
        return []
    df = pd.read_csv('./data/messages.csv').dropna()
    return list(map(lambda data: Email(data[0], data[1]), zip(df['subject']
                                    + ' ' + df['message'], df['label'])))

@staticmethod
def readTrainTestSplit():
    data = Email.read()
    df = pd.DataFrame()
    df['message'] = [email.content for email in data]
    f = feature_extraction.text.CountVectorizer()
    df['label'] = [email.label for email in data]
    return train_test_split(f.fit_transform(df['message']),
            df['label'], test_size=0.2, random_state=225, stratify=df['label'])
```

Chuyển email về chữ thường

```
self.content = self.content.lower()
```

Loại bỏ các từ không mang ý nghĩa trong câu (stopwords)

Các tử không mang ý nghĩa trong câu như will, have, ... cần được loại bỏ.

```
for word in stopwords.words('english'):
    self.content = re.sub(' ' + word + ' ',  ' ', self.content)
```

Thay thế các đường link, số, địa chỉ email

```
self.content = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'MailID', self.content)
self.content = re.sub(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
'Links', self.content)
self.content = re.sub(r'\d+(\.\d+)?', 'numbers', self.content)
```

Loại bỏ các kí tự đặc biệt.

```
self.content = re.sub(r'\n', ' ', self.content)
self.content = re.sub(r"[^a-zA-Z0-9]+", " ", self.content)
```

Kết quả

Để thấy rõ được kết quả của quá trình tiền xử lý email có một ví dụ cụ thể như sau.
Trước khi xử lý

```
"""job posting - apple-iss research center content - length : 3386 apple-iss
research center a us $ 10 million joint venture between apple computer inc . and
the institute of systems science of the national university of singapore ,
located in singapore , is looking for : a senior speech scientist - - - - - - -
- - - - - - - - - - - - - - - - - - - - the successful candidate will have research
expertise in computational linguistics , including natural language processing
and * * english * * and * * chinese * * statistical language modeling .
knowledge of state-of - the-art corpus-based n - gram language models , cache
language models , and part-of - speech language models are required . a text -
to - speech project leader - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - the successful candidate will have research expertise expertise in two
or more of the following areas : computational linguistics , including natural
language parsing , lexical database design , and statistical language modeling ;
text tokenization and normalization ; prosodic analysis . substantial knowledge
of the phonology , syntax , and semantics of chinese is required . knowledge of
acoustic phonetics and / or speech signal processing is desirable . both
candidates will have a phd with at least 2 to 4 years of relevant work
experience , or a technical msc degree with at least 5 to 7 years of experienc e
. very strong software engineering skills , including design and implementation
, and productization are required in these positions . knowledge of c , c + +
and unix are preferred . a unix & c programmer - - - - - - - - - - - - - - - - -
- - - we are looking for an experienced unix & c programmer , preferably with
good industry experience , to join us in breaking new frontiers . strong
knowledge of unix tools ( compilers , linkers , make , x - windows , e - mac , .
. . ) and experience in matlab required . sun and silicon graphic experience is
an advantage . programmers with less than two years industry experience need not
apply . these positions include interaction with scientists in the national
university of singapore , and with apple 's speech research and productization
efforts located in cupertino , california . attendance and publication in
international scientific / engineering conferences is encouraged . benefits
include an internationally competitive salary , housing subsidy , and relocation
expenses . _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ send a complete
resume , enclosing personal particulars , qualifications , experience and
contact telephone number to : mr jean - luc lebrun center manager apple - iss
research center , institute of systems science heng mui keng terrace , singapore
0511 tel : ( 65 ) 772-6571 fax : ( 65 ) 776-4005 email : jllebrun @ iss . nus .
sg"""
```

Sau khi xử lý

```
"""job posting apple iss research center content length numbers apple iss research
center us numbers million joint venture apple computer inc institute systems
science national university singapore located singapore looking senior speech
scientist successful candidate research expertise computational linguistics
including natural language processing english chinese statistical language
modeling knowledge state of the art corpus based n gram language models cache
language models part of speech language models required text speech project
leader successful candidate research expertise expertise two following areas
computational linguistics including natural language parsing lexical database
design statistical language modeling text tokenization normalization prosodic
analysis substantial knowledge phonology syntax semantics chinese required
knowledge acoustic phonetics speech signal processing desirable candidates phd
least numbers numbers years relevant work experience technical msc degree least
numbers numbers years experienc e strong software engineering skills including
design implementation productization required positions knowledge c c unix
preferred unix c programmer looking experienced unix c programmer preferably
good industry experience join us breaking new frontiers strong knowledge unix
tools compilers linkers make x windows e mac experience matlab required sun
silicon graphic experience advantage programmers less two years industry
experience need apply positions include interaction scientists national
university singapore apple s speech research productization efforts located
cupertino california attendance publication international scientific engineering
conferences encouraged benefits include internationally competitive salary
housing subsidy relocation expenses send complete resume enclosing personal
particulars qualifications experience contact telephone number mr jean luc
lebrun center manager apple iss research center institute systems science heng
mui keng terrace singapore numbers tel numbers numbers numbers fax numbers
numbers numbers email jllebrun iss nus sg"""
```

Qua đó có thể thấy email đã không còn chứa nhiều thành phần gây nhiễu nữa góp phần làm tăng độ chính xác cho bộ lọc.

Chia tập dữ liệu thành các tập data/test

```
X_train, X_test, Y_train, Y_test = Email.readTrainTestSplit()
```

3.2 Support vector machine

Ở phần hiện thực này, em xin trình bày thuật toán trên bộ thư viện sklearn.
Trước hết ta khai báo những thư viện cần thiết

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.svm import SVC
```

Ở đây em dùng model C-Support Vector Classification có sẵn ở sklearn library. Với Kernel là 'rbf'. Thực ra đây là soft-margin SVM vì có hằng số C=1.0 là hằng số chịu lỗi.

```
# kernel = 'rbf'
model1 = SVC(C=1.0, kernel='rbf')
model1.fit(X_train,Y_train)
y_pred = model1.predict(X_test)
print("Accuracy : ", accuracy_score(y_pred,Y_test))
print("Precision : ", precision_score(y_pred,Y_test, average ='weighted'))
print("Recall : ", recall_score(y_pred,Y_test, average ='weighted'))
```

Ta được kết quả của phương pháp này là:
Accuracy : 0.9241622574955908
Precision : 0.9588539907688843
Recall : 0.9241622574955908
Vì đây là một trong những phương pháp thường đạt kết quả tốt nhất nên em sử dụng nó như là cách tiếp cận đầu tiên. Ta thấy kết quả rất tốt.

Trong quá trình hiện thử, em đã thử với kernel trick là 'linear' và may mắn thay kết quả được phân loại đúng gần như hoàn toàn.

```
# kernel = 'linear'
model3 = SVC(C=1.0, kernel='linear')
model3.fit(X_train,Y_train)
y_pred = model3.predict(X_test)
print("Accuracy : ", accuracy_score(y_pred,Y_test))
print("Precision : ", precision_score(y_pred,Y_test, average ='weighted'))
print("Recall : ", recall_score(y_pred,Y_test, average ='weighted'))
```

Kết quả:
Accuracy : 0.9947089947089947
Precision : 0.9946976896506833
Recall : 0.9947089947089947

Trong thực tế thì kernel trick 'rbf' thường tốt hơn so với 'linear' nhưng theo cách trình bày ở phần lý thuyết ta thấy, rất khó để khẳng định phương pháp nào tốt hơn hoàn toàn bởi vì mỗi kernel nó có một mục đích phân loại riêng và tùy vào dữ liệu đầu vào. Ở đây phần làm sạch dữ liệu nhóm em làm rất tốt kết hợp với phương pháp chuyển đổi truyền thống tf-idf thì kết quả cả 2 cách chọn kernel trick đều rất tốt.

3.3 Naive Bayes

Ở đây ta dùng Multinomial Naive Bayes cho phân loại thư.

Đầu tiên import các thư viện cần thiết:

```
import numpy as np
from sklearn import feature_extraction, model_selection, naive_bayes, metrics
```

Chúng ta sẽ huấn luyện model với các tham số α khác nhau để tìm tham số phù hợp cho kết quả tốt nhất cho mô hình.

```
# Chọn alpha
list_alpha = np.arange(1/100000, 20, 0.1)
score_train = np.zeros(len(list_alpha))
score_test = np.zeros(len(list_alpha))
recall_test = np.zeros(len(list_alpha))
precision_test= np.zeros(len(list_alpha))
count = 0
for alpha in list_alpha:
    bayes = naive_bayes.MultinomialNB(alpha=alpha)
    bayes.fit(X_train, y_train)
    score_train[count] = bayes.score(X_train, y_train)
    score_test[count]= bayes.score(X_test, y_test)
    recall_test[count] = metrics.recall_score(y_test, bayes.predict(X_test))
    precision_test[count] = metrics.precision_score(y_test, bayes.predict(X_test))
    count = count + 1

matrix = np.matrix(np.c_[list_alpha, score_train,
score_test, recall_test, precision_test])
models = pd.DataFrame(data = matrix, columns =
['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
models.head(n=10)
```

Kết quả của 10 giá trị α đầu tiên:

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	1.000000	0.991182	0.94382	1.000000
1	0.10001	0.998675	0.998236	1.00000	0.988889
2	0.20001	0.997792	0.994709	1.00000	0.967391
3	0.30001	0.997350	0.994709	1.00000	0.967391
4	0.40001	0.997350	0.994709	1.00000	0.967391
5	0.50001	0.997350	0.994709	1.00000	0.967391
6	0.60001	0.997350	0.994709	1.00000	0.967391
7	0.70001	0.997350	0.994709	1.00000	0.967391
8	0.80001	0.997350	0.994709	1.00000	0.967391
9	0.90001	0.997350	0.994709	1.00000	0.967391

Tiếp theo ta sẽ chọn giá trị α mà mô hình tốt nhất. Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đó hiệu quả thường được sử dụng là Precision-Recall. Ở đây ta ưu tiên chọn mô hình theo test precision mà không phải là train precision để tránh tình trạng có thể xảy ra overfit trong tập train.

```
best_index = models['Test Precision'].idxmax()
models.iloc[best_index, :]
```

Kết quả tham số α cho kết quả tốt nhất là khoảng 0.00001

```
alpha              0.000010
Train Accuracy     1.000000
Test Accuracy      0.991182
Test Recall        0.943820
Test Precision     1.000000
Name: 0, dtype: float64
```

Trong trường hợp có nhiều tham số α đều cho giá trị test precision tốt nhất thì ta cần ưu tiên chọn tiếp theo test accuracy.

Ở kết quả trên ta thấy giá trị test precision lớn nhất là 1 nên ta tìm thử xem còn tham số nào cho giá trị này không bằng :

```
models[models['Test Precision']==1].head(n=10)
```

Kết quả:

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	1.000000	0.991182	0.943820	1.0
34	3.40001	0.994700	0.992945	0.955056	1.0
35	3.50001	0.994700	0.992945	0.955056	1.0
36	3.60001	0.994258	0.992945	0.955056	1.0
37	3.70001	0.994258	0.992945	0.955056	1.0
38	3.80001	0.994258	0.992945	0.955056	1.0
39	3.90001	0.994258	0.992945	0.955056	1.0
40	4.00001	0.994258	0.992945	0.955056	1.0
41	4.10001	0.994258	0.991182	0.943820	1.0
42	4.20001	0.994700	0.991182	0.943820	1.0

Tiếp tục chọn siêu tham số α theo giá trị test accuracy

```
best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
bayes = naive_bayes.MultinomialNB(alpha=list_alpha[best_index])
bayes.fit(X_train, y_train)
models.iloc[best_index, :]
```

Kết quả:

```
alpha              3.400010
Train Accuracy     0.994700
Test Accuracy      0.992945
Test Recall        0.955056
Test Precision     1.000000
Name: 34, dtype: float64
```

3.3.1 Confusion matrix trong naive bayes

- Sau khi chọn được giá trị α thích hợp và train mô hình xong ta được kết quả phân loại như sau

```
m_confusion_test = metrics.confusion_matrix(y_test, bayes.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
             index = ['Actual 0', 'Actual 1'])
```

Kết quả phân loại như sau (chú ý chỉ đang xét trong tập test):

	Predicted 0	Predicted 1
Actual 0	478	0
Actual 1	4	85

- Nhận xét: trong 567 email ta phân loại được đúng 478 email không rác và đúng 85 email rác trong đó bỏ sót 4 email rác với tỉ lệ chính xác hơn 99%

3.4 Neural network

Đầu tiên ta import một số thư viện cần thiết

```
import tensorflow.keras as keras
from keras.models import  Sequential
from keras.layers import Dense ,Conv1D,MaxPooling1D,Flatten
```

Tiếp theo ta xác định cấu trúc mạng Neural

Dense là một lớp mức đầu vào do Keras cung cấp, lớp này chấp nhận số lượng nơron hoặc đơn vi làm tham số bắt buộc của nó. Nếu lớp là lớp đầu tiên, thì ta cũng cần cung cấp Hình dạng đầu vào, . Nếu không, đầu ra của lớp trước sẽ được sử dụng làm đầu vào của lớp tiếp theo. Tất cả các thông số khác là tùy chọn.Tham số đầu tiên đại diện cho số lượng đơn vị (nơ-ron)

Trong học máy, tất cả các loại dữ liệu đầu vào như văn bản, hình ảnh hoặc video sẽ được chuyển đổi đầu tiên thành mảng số và sau đó đưa vào thuật toán. Số đầu vào có thể là mảng một chiều, mảng hai chiều (ma trận) hoặc mảng nhiều chiều. Ta có thể chỉ định thông tin về chiều bằng cách sử dụng hình dạng, một bộ số nguyên. Ví dụ, (4,2) đại diện cho ma trận có bốn hàng và hai cột.

```
model = Sequential()
model.add(Dense(256, activation='relu',input_dim=X_train.shape[1]))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Huấn luyện mô hình

```
history = model.fit(X_train, Y_train, validation_split = 0.1,
                            epochs=5, batch_size=32,shuffle=True)
```

```
Epoch 1/5
64/64 [==============================] - 7s 95ms/step - loss: 0.2952 - accuracy: 0.8827 - val_loss: 0.1031 - val_accuracy: 0.9956
Epoch 2/5
64/64 [==============================] - 6s 93ms/step - loss: 0.0275 - accuracy: 0.9985 - val_loss: 0.0250 - val_accuracy: 0.9956
Epoch 3/5
64/64 [==============================] - 6s 93ms/step - loss: 2.7581e-04 - accuracy: 1.0000 - val_loss: 0.0313 - val_accuracy: 0.9912
Epoch 4/5
64/64 [==============================] - 6s 92ms/step - loss: 1.3545e-04 - accuracy: 1.0000 - val_loss: 0.0338 - val_accuracy: 0.9912
Epoch 5/5
64/64 [==============================] - 6s 92ms/step - loss: 8.6173e-05 - accuracy: 1.0000 - val_loss: 0.0370 - val_accuracy: 0.9824
```

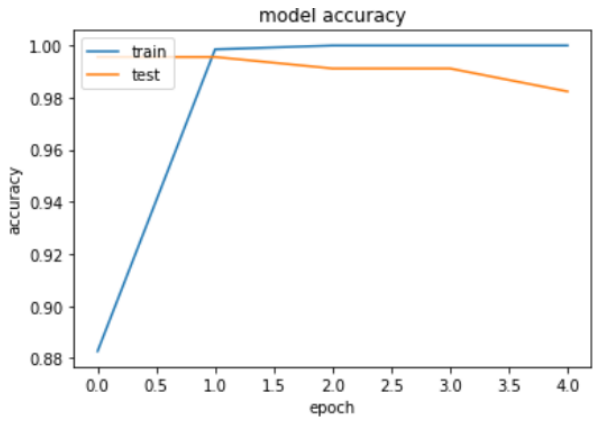
Hình 13: *Mô hình*

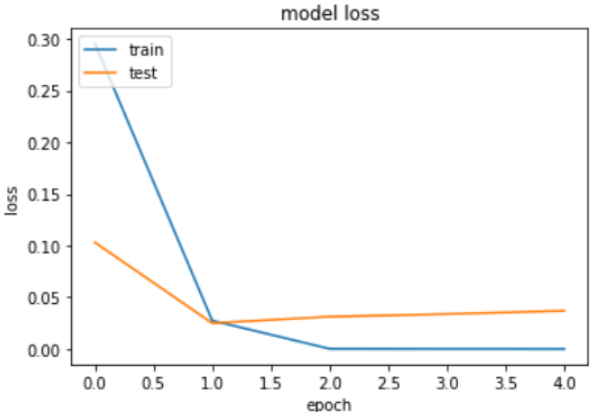
Hiển thị lịch sử đào tạo :

```
# hiển thị lịch sử đào tạo

print(history.history.keys())

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Đánh giá mô hình trên tập test

```
results = model.evaluate(X_test, Y_test, batch_size=128)
print("test loss, test acc:", results)
```

```
results = model.evaluate(X_test, Y_test, batch_size=128)
print("test loss, test acc:", results)
```

```
5/5 [==============================] - 0s 70ms/step - loss: 0.0240 - accuracy: 0.9929
test loss, test acc: [0.0240020751953125, 0.9929453134536743]
```

Hình 14: *Kết quả đánh giá*

Ta thấy mô hình hoạt động khá tốt trên cả tập test
Cuối cùng ta lưu mô hình để khi cần có thể load lên vào sử dụng .

```
model.save('model.h5')
```

Mô hình của chúng ta sau khi train xong đã được lưu trong file model.h5.

4 Đánh giá

Như vậy, qua đề tài này, nhóm đã hiện thực thành công các mô hình Machine Learning, ngoài ra còn bổ sung thêm Neural Network để ứng dụng vào phân loại thư rác và thu được kết quả khả quan, kết quả bao gồm các thông số accuracy, precision và recall có thể được xem ở bảng sau:

	Soft-margin SVM("rbf" kernel trick)	Soft-margin SVM ("linear" kernel trick)	Naive Bayes	Neural Network
Accuracy	0.92416	0.99470	0.992945	0.992947
Precision	0.95885	0.99470	1.0	
Recall	0.92416	0.99470	0.955056	

Ý nghĩa của các giá trị:

- Accuracy: Đây là độ đo của bài toán phân loại mà đơn giản nhất, tính toán bằng cách lấy số dự đoán đúng chia cho toàn bộ các dự đoán.
- Precision: Sẽ có rất nhiều trường hợp thước đo Accuracy không phản ánh đúng hiệu quả của mô hình (accuracy cao nhưng mô hình không tốt). Precision là một trong những metrics có thể khắc phục được, nó đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm positive.
- Recall: Recall cũng là một metric quan trọng, nó đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm true positive và false negative.

Nhìn chung, các mô hình trên thì đều cho kết quả tốt (các giá trị đều cao, hầu hết đều trên 0.95) và khá tương đương với nhau, vì kích thước của tập dữ liệu chỉ nằm ở mức vừa phải. Riêng trường hợp Soft-margin SVM ("linear" kernel trick) thì gặp may mắn nên kết quả gần như tuyệt đối. Trong đề tài, nhóm muốn khẳng định lại rằng, dữ liệu là cực kỳ quan trọng, việc làm sạch và phân tích chúng theo hướng nào ảnh hưởng rất lớn đến thuật toán và không phải thuật toán càng phức tạp và càng tốt về mặt lý thuyết thì cho kết quả tốt ở thực tế.

Qua đề tài lần này, nhóm đã tìm hiểu được cách xây dựng một bài toán giải quyết theo machine learning như thế nào và cách đo đạt kết quả ra sao. Đồng thời tìm hiểu được nhiều thứ về thuật toán và phân tích dữ liệu.

TÀI LIỆU THAM KHẢO

[1] Quang Minh. *Ứng dụng học máy trong lọc thư rác.* http://antoanthongtin.vn/giai-phap-khac/ung-dung-hoc-may-trong-loc-thu-rac-107401

[2] Trà My. *Convolutional Neural Network là gì? Cách chọn tham số cho Convolutional Neural Network chuẩn chỉnh.* https://wiki.tino.org/convolutional-neural-network-la-gi/

[3] *Bài 14: Multi-layer Perceptron và Backpropagation.* https://machinelearningcoban.com/2017/02/24/mlp/

[4] Pham Van Chung *[Deep Learning] Tìm hiểu về mạng tích chập (CNN).* https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2

[5] *[SVM] Bài 19, 20, 21 về Support Vector Machine, Soft Margin SVM, Kernel Support Vector Machine:* https://machinelearningcoban.com/2017/04/09/smv/