TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

----- oOo -----

BÁO CÁO THỰC TẬP KỸ THUẬT

Chuyên ngành: Toán tin

Sinh viên thực hiện: Nguyễn Thị Hằng

MSSV: 20161384

Lớp: Toán Tin - K61

Hà Nội, 2020

LỜI NÓI ĐẦU

Những năm gần đây, trí tuệ nhân tạo cụ thể hơn là máy học nổi lên như bằng chứng cho cuộc cách mạng mạng công nghệ 4.0. Với sự phát triển đột phá trí tuệ nhân tạo đang có mặt hầu hết mọi lĩnh vực và có ứng dụng mạnh mẽ trong thực tiễn như xe tự lái Tesla, hệ thống gợi ý tag bạn bè dựa vào khuôn mặt của facebook, trợ lý ảo google, hệ thống gợi ý phim netfix, gợi ý sản phẩm của amazon, chatbot …

Sự đặc biệt, mới mẻ và mang lại cơ hội việc làm hấp dẫn hứa hẹn sau này từ lĩnh vực trí tuệ nhân tạo đã thu hút sự quan tâm và mong muốn tìm hiểu trải nghiệm của em trong lĩnh vực này. Em đã rất may mắn khi có thể tham gia thực tập kỹ thuật tại công ty iCOMM Media & Tech, Jsc để có thể trải nghiệm môi trường làm việc thực tế, được tham gia training bởi các anh(chị) có chuyên môn cao, được tham gia tìm hiểu các đề tài thú vị có ứng dụng thực tế. Em xin gửi lời cảm ơn đến tất cả anh(chị) đã hướng dẫn giúp đỡ em trong quá trình thực tập tại đây. Và đặc biệt em xin gửi lời cảm ơn đến anh Tuấn đã cho em những lời khuyên, góp ý hữu ích về việc trình bày một báo cáo như thế nào là hiệu quả. Ngoài ra em cũng xin gửi lời cảm ơn đến anh Đạt, thầy Linh, bạn Đàm đã cũng cấp cho em những kiến thức chuyên môn hữu ích trong quá trình training và hoàn thành đề tài được giao.

Em xin gửi lời cảm ơn đến toàn thể thầy cô viện Toán ứng dụng và tin học trường Đại học Bách Khoa Hà Nội đã tạo điều kiện cho em thực tập tại Công ty iCOMM Media & Tech, Jsc. Đồng thời gửi lời cảm ơn đến thầy cô bộ môn đã cung cấp cho em vốn kiến thức nền tảng, giúp quá trình thực tập ở công ty dễ dàng hơn.

Hà nội, tháng 09 năm 2020

Người thực hiện đồ án

Hằng

Nguyễn Thị Hằng

MỤC LỤC

DANH SÁCH HÌNH VẼ

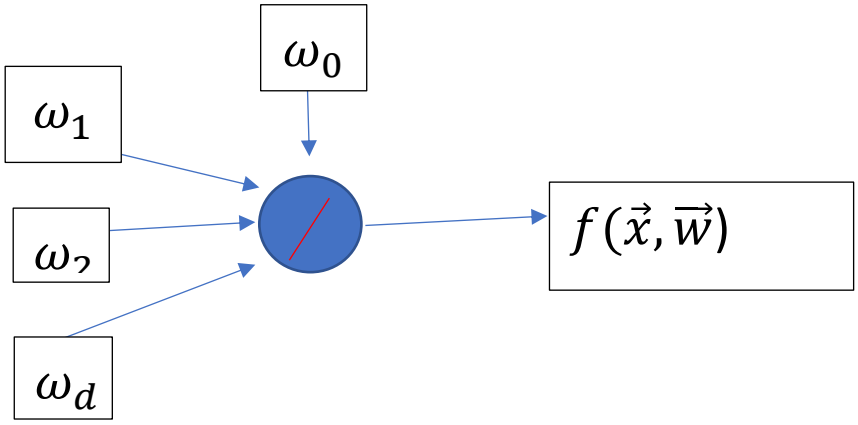
CHƯƠNG 1: NỘI DUNG TRAINING

1.1 Giới thiệu mạng neural nhân tạo

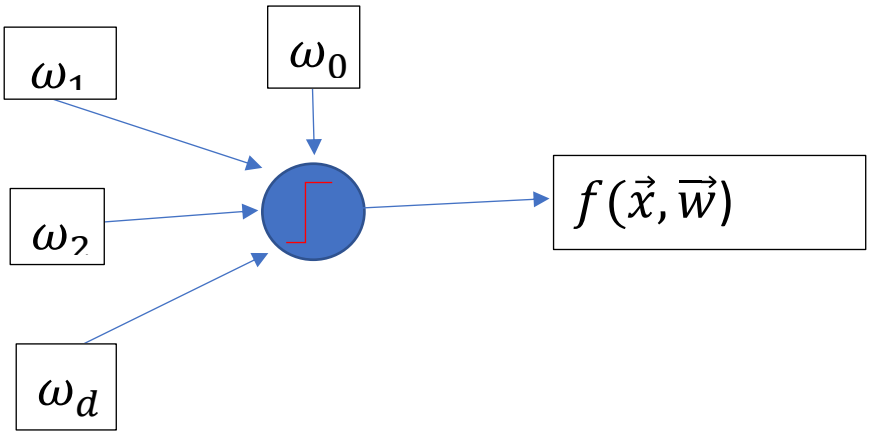
Mạng neural nhân tạo hay thường gọi ngắn gọn mạng neural (tiếng Anh là Artificial Neural network - ANN hay Neural Network) là một mô hình toán học hay mô hình tính toán được sử dụng phổ biến trong lĩnh vực machine learning và deep learning. Nó gồm có một nhóm các neural nhân tạo được nối với nhau thông qua các liên kết (trọng số liên kết) và làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Việc xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút. Trong nhiều trường hợp, mạng neural nhân tạo là một hệ thống thích ứng (*adaptive system*) tự thay đổi cấu trúc của mình dựa trên các thông tin bên ngoài hay bên trong chảy qua mạng trong quá trình học [1].

Một mạng neural nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu…) thông qua một quá trình học từ tập các mẫu huấn luyện. Một mạng neural nhân tạo thường được kết hợp từ ba thành phần: đơn vị xử lý, hàm kết hợp, hàm kích hoạt [2].

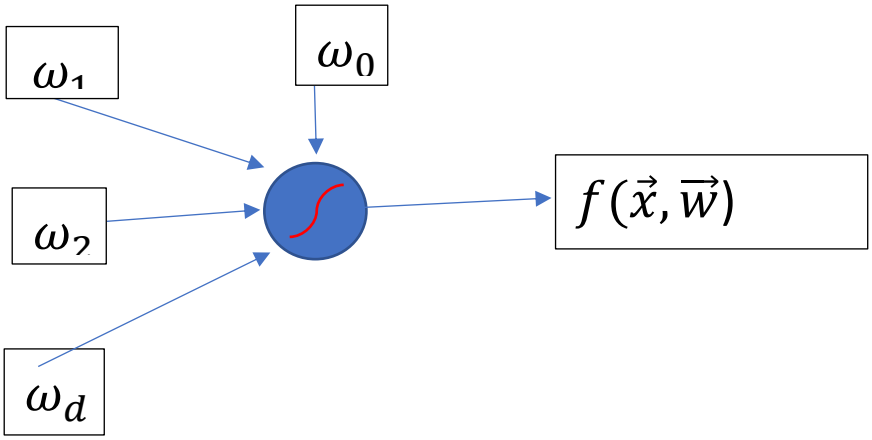
Ví dụ một số mạng neural cơ bản:



Hình 1 Mô hình mạng neural tuyến tính



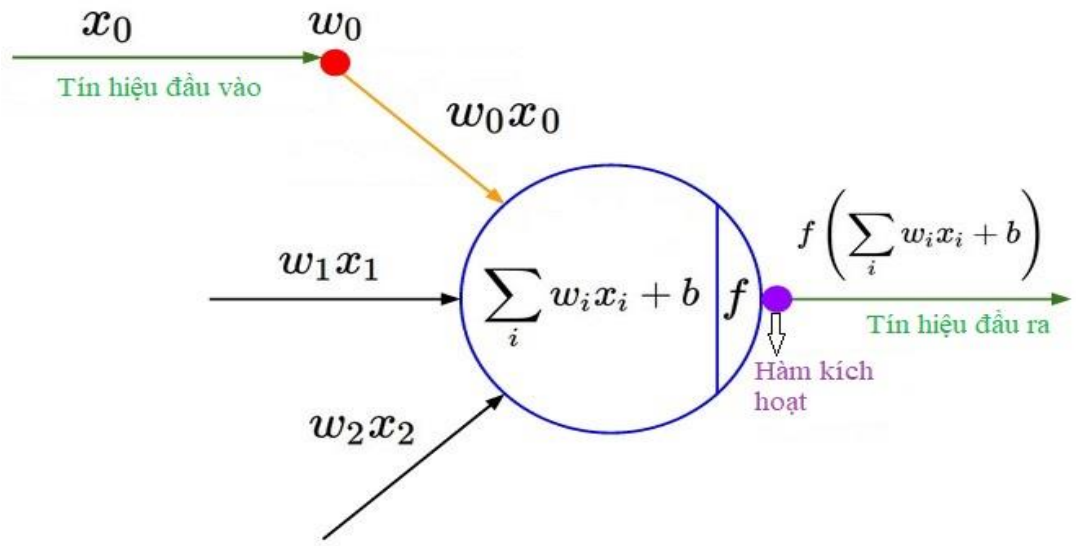
Hình 2 Mô hình mạng neural perceptron



Hình 3 Mô hình mạng neural sigmoid

Đơn vị xử lý

Còn được gọi là một neural hay một nút (node), thực hiện một công việc rất đơn giản: Nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu đầu ra sẽ được làn truyền sang một đơn vị khác. Cấu trúc của một neural được minh họa như sau:



Hình 4 Mô hình chung của mạng neural

Trong đó:

- ✓ X_i là các tín hiệu đầu vào
- ✓ W_i là các trọng số. Tương ứng với X_i có trọng số W_iX_i
- ✓ b là hệ số bias
- ✓ $a_i = \sum_i w_i x_i + b$ là đầu vào mạng (net input)
- ✓ $z_i = f(\sum_i w_i x_i + b)$ là đầu ra của neural
- ✓ f hàm kích hoạt

Một mạng neural có ba kiểu đơn vị:

- ✓ Các đơn vị đầu vào (input units), nhận tín hiệu từ bên ngoài
- ✓ Các đơn vị đầu ra (output units), gửi tín hiệu ra bên ngoài
- ✓ Các đơn vị ẩn (hidden units) các tín hiệu vào (input) và ra (output) của nó nằm trong mạng

Mỗi đơn vị *i* có thể có một hoặc nhiều đầu vào nhưng chỉ có một đầu ra. Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc là đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

Hàm kết hợp:

Mỗi một đơn vị trong mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi luật lan truyền cụ thể. Trong phần lớn các mạng neural, chúng ta giả sử rằng mỗi một đơn vị cung cấp một bộ cộng như là đầu vào cho đơn vị mà nó có liên kết. tổng đầu vào đơn vị *i* đơn giản chỉ là trọng số của các đầu ra riêng lẻ từ các đơn vị kết nối cộng thêm ngưỡng hay độ lệch *b (bias):*

$$a_i = \sum_i w_i x_i + b$$

Trường hợp $w_i > 0$, neural được coi là đang ở trong trạng thái kích thích. tương tự nếu như $w_i < 0$, neural ở trạng thái kiềm chế.

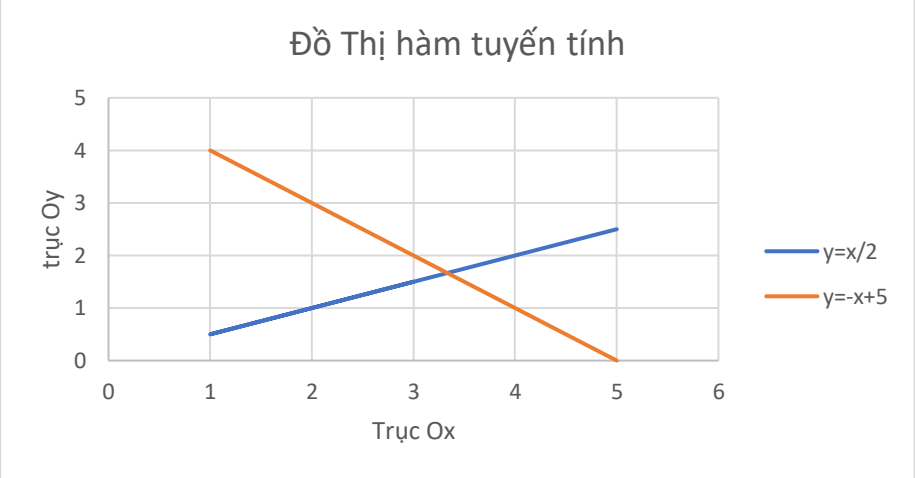
Hàm kích hoạt:

Phần lớn các đơn vị trong mạng neural chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt [3], kết quả của hàm này là một giá trị gọi là mức kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm ép (squashing). Các hàm kích hoạt được sử dụng là:

Hàm đồng nhất (Linear function , Identity function):

$$l(x) = x$$

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



Hình 5 Đồ thị hàm tuyến tính

Công thức hàm mất mát:

$$J = \frac{1}{2} * \frac{1}{N} * \left(\sum_{i=1}^{N} (\hat{y_i} - y_i)^2 \right)$$

$$\hat{y_i} = \omega_i * x_i + \omega_0$$

- ✓ J không âm
- ✓ J càng nhỏ thì đường thẳng càng gần điểm dữ liệu. Nếu j=0 thì đường thẳng đi qua tất cả các điểm dữ liệu

  => Bài toán đặt ra là cực tiểu hàm mất mát để thu được kết quả tốt nhất có thể

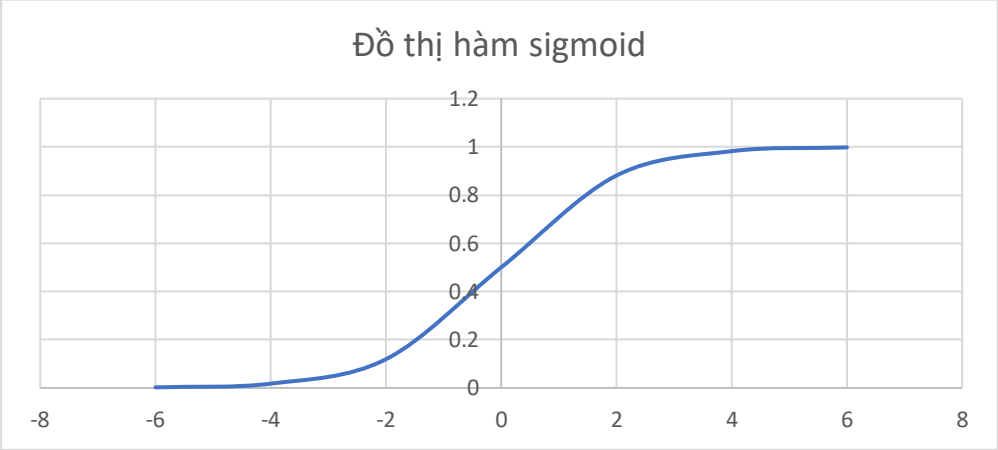
Hàm sigmoid (Sigmoid function (logsig)) [4]**:**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Quan hệ linear function với logistic function:

$$\hat{y} = \sigma\left(\omega_0 + \omega_1 * x_1^{(i)} + \omega_2 * x_2^{(i)}\right) = \frac{1}{1 + e^{-(\omega_0 + \omega_1 * x_1^{(i)} + \omega_2 * x_2^{(i)})}}$$

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện bởi thuật toán Lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng (0,1).



Hình 6 Đồ thị hàm Sigmoid

Hàm Sigmoid bão hào và triệt tiêu gradient: Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương) thì giá trị hàm mất mát sẽ không thay đổi, gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với đơn vị đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).

Đạo hàm logistic function:

Hàm logistic có đạo hàm tại mọi điểm và dễ tính toán:

$$\frac{d}{dx}f(x) = f(x)(1 - f(x))$$

Công thức hàm mất mát - Loss function:

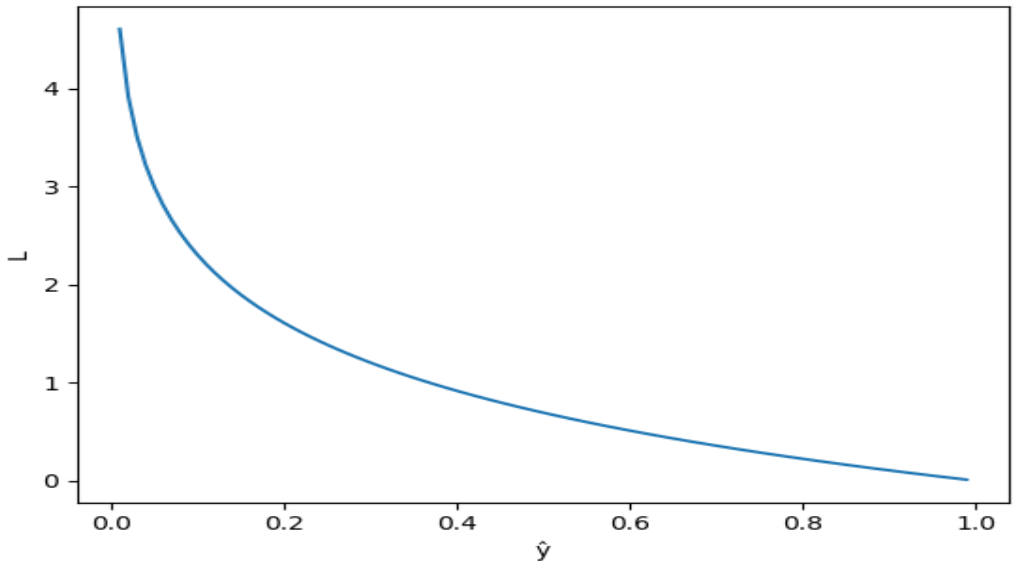
Đây là hàm đánh giá độ tốt của mô hình. Như vậy $\hat{y}$ càng gần y càng tốt:

$$L = -(y_i * \log(\hat{y_i}) + (1 - y_i) * \log(1 - \hat{y_i}))$$

($Chú\ ý\ đây\ là\ trường\ hợp\ đặc\ biệt\ của\ hàm\ mất\ mát\ cross\ entropy$)

Đánh giá L nếu:

$$y_i = 1 => L = -\log(\hat{y_i})$$

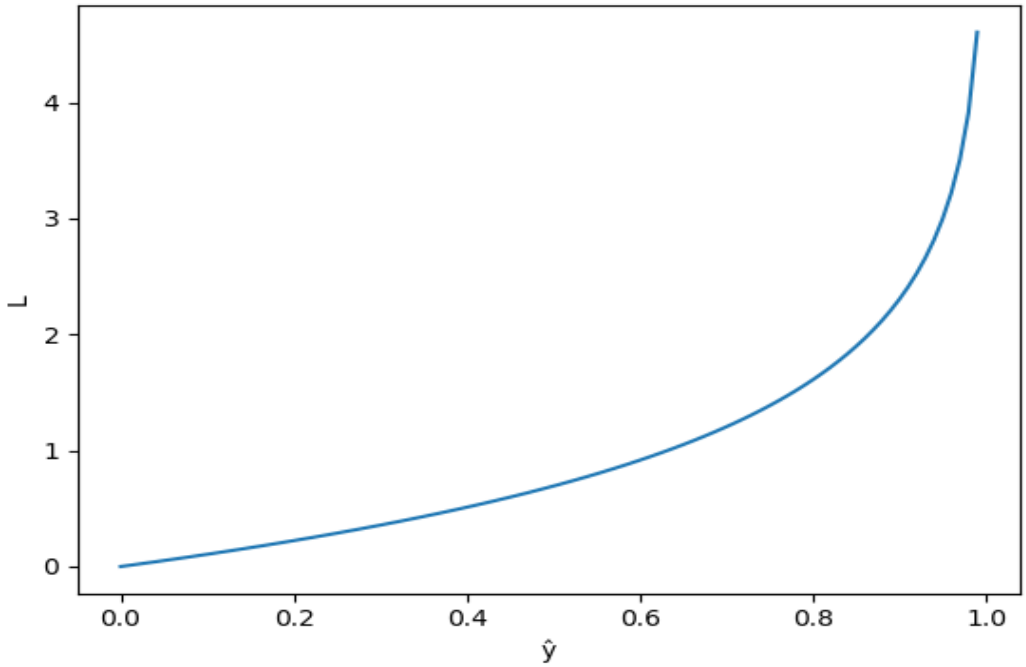


Hình 7 Đồ thị loss function – logistic (trường hợp: y_i=1)

- ✓ Hàm L giảm dần từ 0 đến 1.
- ✓ Khi mô hình dự đoán $\hat{y_i}$ gần 1, tức giá trị dự đoán gần với giá trị thật y_i thì L nhỏ, xấp xỉ 0.
- ✓ Khi mô hình dự đoán $\hat{y_i}$ gần 0, tức giá trị dự đoán ngược lại giá trị thật y_i thì L rất lớn.

Ngược lại nếu:

$$y_i = 0 => L = -\log(1 - \hat{y}_i)$$

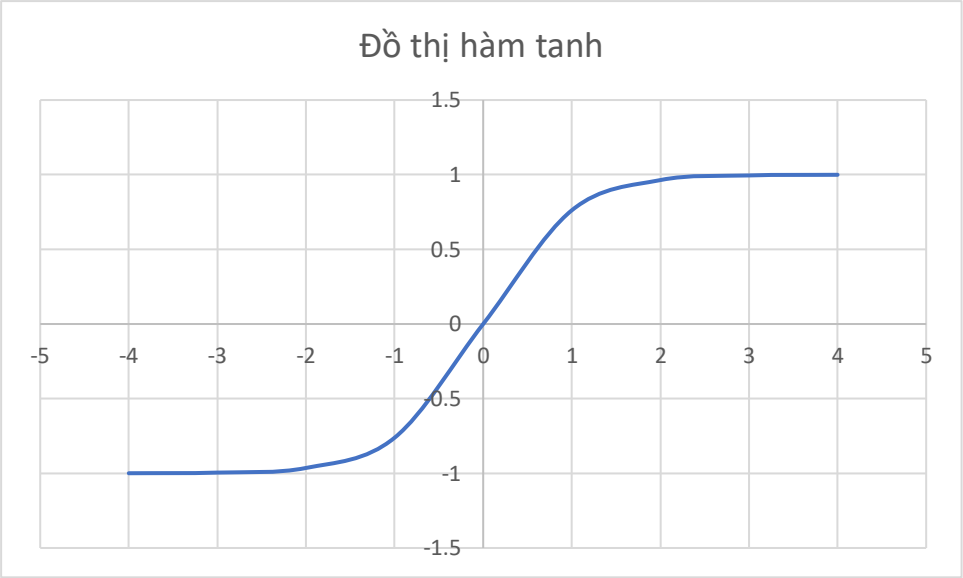


Hình 8 Đồ thị loss function – logistic (trường hợp y_i=0)

- ✓ Hàm L tăng dần từ 0 đến 1.
- ✓ Khi mô hình dự đoán $\hat{y}_i$ gần 0, tức giá trị dự đoán gần với giá trị thật y_i thì L nhỏ, xấp xỉ 0.
- ✓ Khi mô hình dự đoán $\hat{y}_i$ gần 1, tức giá trị dự đoán ngược lại giá trị thật y_i thì L rất lớn.

Hàm tanh (Tanh function):

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



Hình 9 Đồ thị hàm tanh

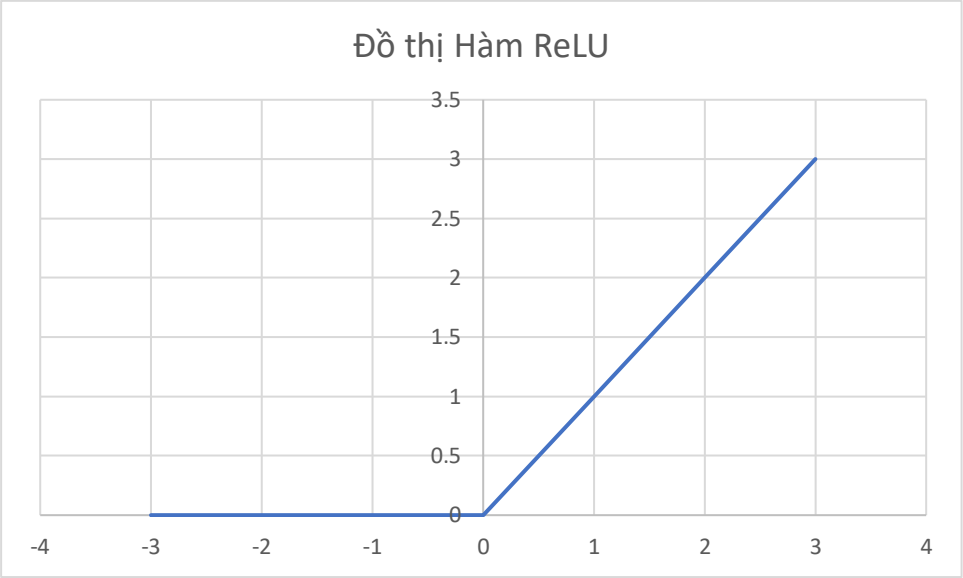
Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1), tương tự hàm Sigmoid lưỡng cực. Cũng như Sigmoid, hàm Tanh bị bão hoà ở hai đầu (gradient thay đổi rất ít ở hai đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được nhược điểm của Sigmoid (khó khăn trong việc hội tụ).

Hàm tanh con được biểu diễn bằng hàm sigmoid như sau:

$$\tanh(x) = \sigma(2x) - 1$$

Hàm ReLu (ReLU function):

$$f(x) = \max(0, x)$$

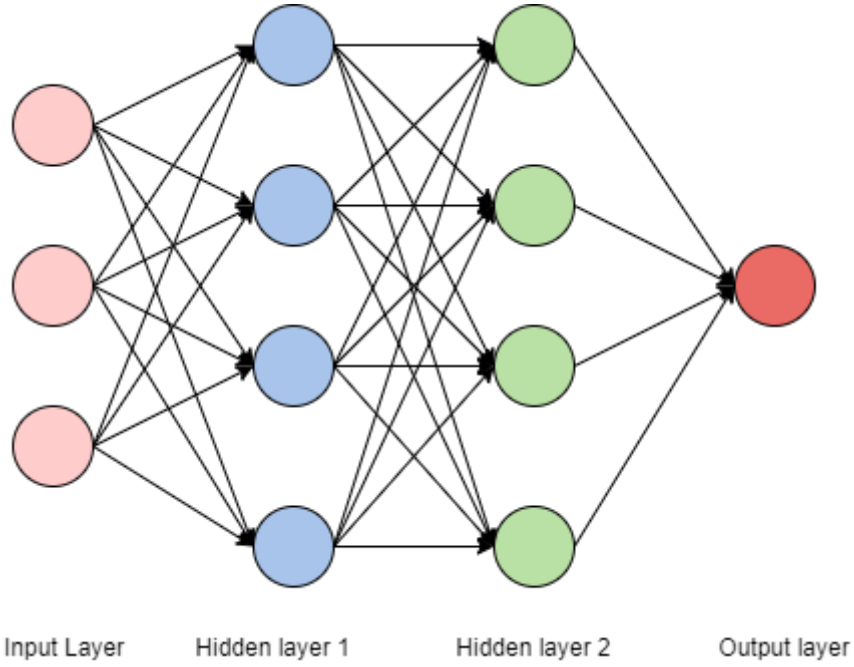


Hình 10 Đồ Thị Hàm ReLU

Hàm ReLU đang được sử dụng lọc các giá trị nhỏ hơn không.

- ✓ **(+)** ReLU có tốc độ hội tụ nhanh điều này do ReLU không bị bão hoà ở 2 đầu.
- ✓ **(+)** Tính toán nhanh hơn, công thức đơn giản tiết kiệm chi phí tính toán.
- ✓ **(-)** Tuy nhiên ReLU cũng có một nhược điểm: Với các nút có giá trị nhỏ hơn không, qua ReLU activation sẽ thành không, hiện tượng đấy gọi là "Dying ReLU". Nếu các nút bị chuyển thành không thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ nút đấy cũng không được cập nhật với gradient descent.
- ✓ **(-)** Khi tốc độ học - learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neural dừng việc cập nhật.

Mạng lan truyền tiến – Feed-Forward Neural Networks (FFNNs): Gồm các mạng perceptron một lớp, mạng perceptron nhiều tầng và mạng RBF [4].



Hình 11 Mô hình mạng lan truyền tiến

Như bạn thấy thì tất cả các nút mạng được kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào tới tầng ra. Tức là mỗi nút ở một tầng nào đó sẽ nhận đầu vào là tất cả các nút ở tầng trước đó mà không suy luận ngược lại. Hay nói cách khác, việc lan truyền trong mạng neural network là lan truyền tiến (feed-forward):

$$z_i^{(l+1)} = \sum_{j=1}^{n^l} w_{ij}^{(l+1)} a_j^{(l)} + b_i^{(l+1)}$$

$$a_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

Trong đó, n^l số lượng nút ở tầng *l* tương ứng và $a_j^{(l)}$ là nút mạng thứ j của tầng *l*. Còn $w_{ij}^{(l+1)}$ là tham số trọng lượng của đầu vào $a_j^{(l)}$ đối với nút mạng thứ *i* của tầng *l+1* và $b_i^{(l+1)}$ là độ lệch (*bias*) của nút mạng thứ *i* của tầng *l+1*. Đầu ra của nút mạng này được biểu diễn bằng $a_j^{(l+1)}$ ứng với hàm kích hoạt $f(z_i)$ tương ứng.

Riêng với tầng vào, thông thường $a^{(l)}$ cũng chính là các đầu vào X tương ứng của mạng.

Để tiện tính toán, ta coi $a_0^{(l)}$ là một đầu vào và $w_{i0}^{(l+1)} = b_i^{(l+1)}$ là tham số trọng lượng của đầu vào này. Lúc đó ta có thể viết lại công thức trên dưới dạng vector:

$$z_i^{(l+1)} = w_i^{(l+1)} . a^{(l)}$$

$$a_i^{(l+1)} = f(z_i^{(l+1)})$$

Nếu nhóm các tham số của mỗi tầng thành một ma trận có các cột tương ứng với tham số mỗi nút mạng thì ta có thể tính toán cho toàn bộ các nút trong một tầng bằng vector:

$$z^{(l+1)} = w^{(l+1)} . a^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)})$$

Một số độ đo đánh giá mô hình:

Đánh giá mô hình:		Dự đoán	
		Âm tính	Dương tính
Thực tế	Âm tính	Âm tính thật	Dương tính giả
	Dương tính	Âm tính giả	Dương tính thật

$$Precision = \frac{Dương\ tính\ thật}{Dương\ tính\ thật + Dương\ tính\ giả}$$

$$Recall = \frac{Dương\ tính\ thật}{Dương\ tính\ thật + Âm\ tính\ giả}$$

$$Chỉ\ số\ F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

1.2 Bài toán phân loại văn bản

Input: bộ data của công ty đã được làm sạch và phân cụm sẵn, gồm 45 category, 71963 văn bản.

Output:

Bài toán sử dụng thuật toán naïve bayes, SVM thì sẽ coi category là nhãn => bài toán phân loại văn bản thông thường.

Bài toán sử dụng K-Means (coi là bài toán unsupervised) xáo trộn data => phân cụm.

1.2.1 Mô hình Naïve Bayes - Naïve Bayes Classifier:

Xét bài toán classification, phân lớp tập dạng $\Omega = \{X \in R^d \mid d \in N\}$ thành C lớp với C = $\{C_i, i = \overline{1, c}\}$ là một phân hoạch của tập dạng Ω. Khi đó cho điểm dữ liệu $X \in \Omega$, việc tính xác suất để điểm dữ liệu rơi vào lớp C_i dựa vào công thức bayes [5]:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Trong đó: $P(X) = \sum_{i=1}^{c} P(X|C_i)P(C_i)$ (Công thức xác suất đầy đủ).

Biểu thức trên nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi lớp. Từ đó có thể giúp xác định lớp của điểm dữ liệu đó bằng cách chọn ra lớp có xác suất cao nhất:

$$c = argmax\big(P(C_i|X)\big), i = \overline{1,c}$$

$$= argmax\left(\frac{P(X|C_i)P(C_i)}{P(X)}\right)$$

$$= argmax(P(X|C_i)P(C_i))$$

(Vì $P(X)$ là một hằng số không phụ thuộc vào C_i nên trong quá trình tính toán có thể bỏ qua $P(X)$).

- ✓ $P(C_i)$ là tần suất điểm dữ liệu trong tập huấn luyện rơi vào lớp C_i, hay còn được tính bằng tỉ lệ số điểm dữ liệu trong tập huấn luyện rơi vào lớp C_i chia cho tổng số điểm dữ liệu trong tập huấn luyện.
- ✓ Việc tính toán $P(X|C_i)$ thường khá là phức tạp nên để đơn giản hóa việc tính toán người ta thường coi các thuộc tính của vector X là độc lập tuyến tính khi đó ta có công thức:

  $$P(X|C_i) = \prod_{k=1}^{d} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i) \dots P(x_d|C_i)$$

Việc tính toán $P(x_k|C_i), k = \overline{1,d}; i = \overline{1,c}$ phụ thuộc vào loại dữ liệu, thông thường sẽ có ba quy tắc phân lớp thường dùng phổ biến: Gaussian Naïve Bayes, Multinomial Naïve Bayes, và Bernoulli Naïve Bayes.

Quy tắc phân lớp Gaussian Naive Bayes [6]:

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(X|C_i) = g(x_k, \mu_{c_i}, \sigma_{c_i})$$

Bộ tham số $\theta = (\mu_{c_i}, \sigma_{c_i})$ được xác định là kỳ vọng μ_{c_i} và pương sai $\sigma_{c_i}^2$ của các thuộc tính x_k tương ứng.

Tuy nhiên trong thư viện sklearn bộ tham số $\theta = (\mu_{c_i}, \sigma_{c_i})$ được tính toán dựa trên công thức:

$$\theta = (\mu_{c_i}, \sigma_{c_i}) = argmax \prod_{k=1}^{d} P(x_k^{(c_i)} | \mu_{c_i}, \sigma_{c_i})$$

$$i = \overline{1, c}$$

Quy tắc phân lớp Multinomial Naïve Bayes [6]:

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà vector đặc trưng được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ k trong mỗi vector chính là số lần từ thứ k xuất hiện trong văn bản đó.

Khi đó, $P(x_k|C_i)$ tỉ lệ với tần suất từ thứ k (hay feature thứ k cho trường hợp tổng quát) xuất hiện trong các văn bản của lớp C_i. Giá trị này có thể được tính bằng cách:

$$\lambda_{c_i}^k = P(x_k|C_i) = \frac{N_{c_i}^k}{N_{c_i}}$$

Trong đó:

- ✓ $N_{c_i}^k$ là tổng số từ thứ k xuất hiện trong lớp C_i, nó còn được tính bằng tổng các thành phần thứ k của các vector đặc trưng ứng với lớp C_i.
- ✓ N_{c_i} là tổng số từ xuất hiện trong lớp C_i kể cả lặp.

$$N_{c_i} = \sum_{k=1}^{d} N_{c_i}^k \Rightarrow \sum_{k=1}^{d} \lambda_{c_i}^k = 1$$

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong lớp C_i thì giá trị $\lambda_{c_i}^k = 0$, điều này dẫn đến $P(X|C_i) = 0$ bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác, để giải quyết việc này, một kỹ thuật được gọi là *Laplace smoothing* được áp dụng:

$$\widehat{\lambda_{c_i}^k} = \frac{N_{c_i}^k + \alpha}{N_{c_i} + d\alpha}$$

Với α là một số dương tùy ý, thông thường $\alpha = 1$, để tránh trường hợp $N_{c_i}^k = 0$, và mẫu số được cộng thêm một giá trị $d\alpha$ để đảm bảo $\sum_{k=1}^{d} \widehat{\lambda_{c_i}^k} = 1$. Như vậy mỗi lớp C_i sẽ được mô tả bằng bộ các số dương có tổng bằng 1:

$$\widehat{\lambda_{c_i}} = \left\{\widehat{\lambda_{c_i}^1}, \widehat{\lambda_{c_i}^2}, \dots, \widehat{\lambda_{c_i}^d}\right\}$$

Quy tắc phân lớp Bernoulli Naïve Bayes [6]:

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. Khí đó $P(x_k|C_i)$ được tính bằng công thức:

$$P(x_k|C_i) = P(k|C_i)^{x_k}(1 - P(k|C_i))^{1-x_k}$$

Trong đó $P(k|C_i)$ xác suất từ thứ k xuất hiện trong các văn bản của lớp C_i.

Xây dựng mô hình Naïve Bayes

❖ Túi từ - Bag of Words (BOW)

Mô hình thường dùng trong các tác vụ phân lớp văn bản (Text Classification). Thông tin sẽ được biểu diễn thành tập các từ đi kèm với tần xuất xuất hiện của mỗi từ này trong văn bản. Bag of Words được dùng như đặc trưng để huấn luyện cho bài toán phân lớp.

Đoạn văn	Phân tích	
Đại học bách khoa Hà Nội là trường đại học hàng đầu Việt Nam về khoa học kỹ thuật.	Đại	2
	Học	3
	…	…
	Thuật	1

Bảng mô tả túi từ

Các bước xây dựng mô hình túi từ:

- ✓ Bước 1: Thu thập dữ liệu và xử lý dữ liệu

  Xây dựng một mảng chứa toàn bộ dữ liệu văn bản trong tập huấn luyện. Các dữ liệu thu thập được sẽ được đem tiền xử lý (các bước tiền xử lý phần thiết kế dữ liệu huấn luyện mạng)
- ✓ Bước 2: Xây dựng bộ từ điển

  Trong bước này ta sẽ xây dựng một bộ danh sách các từ có xuất hiện trong mảng thu thập ở bước 1. Trong đó các từ xuất hiện trong từ điển là duy nhất.
- ✓ Bước 3: Tạo vector cho câu

Bước tiếp theo là chấm điểm các từ trong mỗi tài liệu. Mục tiêu là biến mỗi tài liệu văn bản thành một vectơ mà chúng ta có thể sử dụng làm đầu vào cho một mô hình học máy. Sử dụng thứ tự các từ trong bộ từ điển, chúng ta có thể chuyển đoạn văn bản thành một vectơ nhị phân (có n thành phần với mỗi thành phần có thể nhận giá trị 0 nếu từ không xuất hiện trong văn bản, hoặc nhận giá trị 1 nếu từ đó có xuất hiện trong văn bản) hoặc chuyển thành một vector có n thành phần với các thành phần là biểu thị tần suất xuất hiện của từ.

❖ Tf-idf

✓ Tf

$$tf_i = \frac{n_i}{N_i}$$

Trong đó: $i = \overline{1,n}$,

n là tổng số văn bản trong tập dữ liệu huấn luyện,

n_i tần suất xuất hiện của từ trong văn bản i,

N_i tổng số từ trong văn bản i.

✓ Idf

$$idf_i = log\frac{n}{d}$$

Trong đó: $i = \overline{1,n}$,

n là tổng số văn bản trong tập dữ liệu huấn luyện,

d số văn bản có sự xuất hiện của từ.

✓ Tf-idf

Tf-idf thể hiện trọng số của mỗi từ theo ngữ cảnh văn bản. tf-idf sẽ có giá trị tăng tỷ lệ thuận với số lần xuất hiện của từ trong văn bản và số văn bản có chứa từ đó trên toàn bộ tập tài liệu.

$$tfidf_i = tf_i \text{ x } idf_i$$

Xây dựng mô hình Naïve Bayes ta sẽ thực hiện vector hóa các câu đánh giá, bình luận dựa trên kỹ thuật BOW + Tf-idf và mỗi câu tương ứng với mỗi điểm dữ liệu là vector có d thành phần, tương ứng với số từ trong bộ từ điển. Trong bài báo cáo này, sử dụng quy tắc phân lớp Multinomial Naïve Bayes để tính toán phân lớp điểm dữ liệu.

1.2.2 Mô hình Support Vector Machine (SVM)

Khoảng cách từ một điểm tới một siêu phẳng

Trong không gian 2 chiều, khoảng cách từ một điểm có toạ độ (x_0, y_0) tới đường thẳng có phương trình $\omega_0 + \omega_1 x_1 + \omega_2 x_2 = 0$ được xác định bởi:

$$\frac{|\omega_0 + \omega_1 x_0 + \omega_2 y_0|}{\sqrt{\omega_1^2 + \omega_2^2}}$$

Trong không gian 3 chiều, khoảng cách từ một điểm có toạ độ (x_0, y_0, z_0) tới một mặt phẳng có phương trình $\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 = 0$ được xác định bởi:

$$\frac{|\omega_0 + \omega_1 x_0 + \omega_2 y_0 + \omega_3 z_0|}{\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}}$$

Hơn nữa, nếu bỏ dấu trị tuyệt đối ở tử số, chúng ta có thể xác định được điểm đó nằm về phía nào của *đường thẳng* hay *mặt phẳng* đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu dương nằm về cùng một phía (có thể gọi đây là *phía dương*), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (có thể gọi đây là *phía âm*). Những điểm nằm trên *đường thẳng* hoặc *măt phẳng* sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.

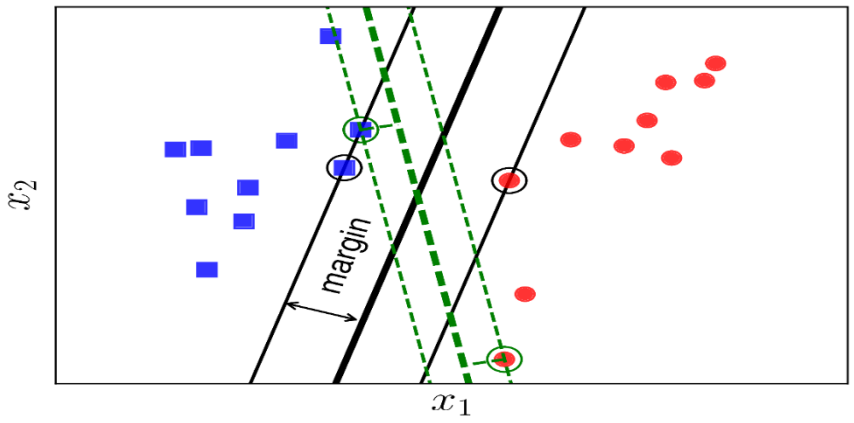
Tổng quát, với không gian n chiều ta có khoảng cách từ một điểm (vector) có tọa độ X_0 đến siêu phẳng có phương trình $\omega_0 + W^T X = 0$ được xác định bởi:

$$\frac{|\omega_0 + W^T X_0|}{||W||_2}$$

Trong đó: $\left|||W|\right|_2 = \sqrt{\sum_{i=1}^{n} \omega_i^2}$ là chuẩn của W theo trọng hàm $u = 2$

Mô hình phân chia hai lớp sử dụng Support Vector Machine

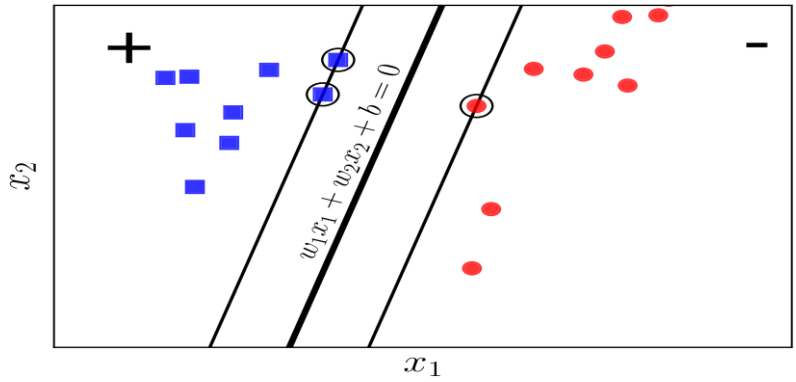
Giả sử rằng có hai lớp khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai lớp này là phân biệt tuyến tính, tức tồn tại một siêu phẳng phân chia chính xác hai lớp đó. Việc tìm một siêu mặt phẳng phân chia hai lớp đó, sao cho tất cả các điểm thuộc một lớp nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc lớp còn lại. (Hoặc một lớp nằm trọn vẹn về phía dương của siêu phẳng hoặc nằm trọn vẹn về phía âm của siêu phẳng).



Hình 12 Lề của hai classes là bằng nhau và lớn nhất có thể

Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi lớp (các điểm được khoanh tròn) tới đường phân chia là như nhau, khoảng cách như nhau này được gọi là *Lề (Margin)*. Lề của cả hai lớp càng lớn thì việc phân lớp càng tốt (càng rạch ròi).

Bài toán tối ưu trong *Support Vector Machine* (SVM) giúp giải quyết vấn đề tìm ra đường phân chia sao cho lề của hai lớp là lớn nhất [7].



Hình 13 Phân tích bài toán SVM.

Các cặp dữ liệu trong tập huấn luyện là (X_1, y_1); (X_2, y_2); …; (X_n, y_n); với vector (X_i) thể hiện là điểm dữ liệu trong tập huấn luyện và (y_i) thể hiện nhãn của điểm dữ liệu đó.

Khoảng cách từ điểm $(X_i, y_i), \forall i \in [1, n]$ tới mặt phân chia có phương trình $\omega_0 + W^T X = 0$ được xác định bởi:

$$\frac{y_i(\omega_0 + W^T X_i)}{||W||_2}$$

Ta có y_i luôn cùng dấu với phía của X_i, như vậy y_i luôn cùng dấu với $\omega_0 + W^T X_i$ và giá trị biểu thức $y_i(\omega_0 + W^T X_i) > 0$. [8] [7]

Với mặt phân chia như trên, lề được tính bằng khoảng cách gần nhất từ một điểm dữ liệu tới mặt phân chia đó (điểm dữ liệu được chọn là điểm bất kỳ thuộc một trong hai lớp):

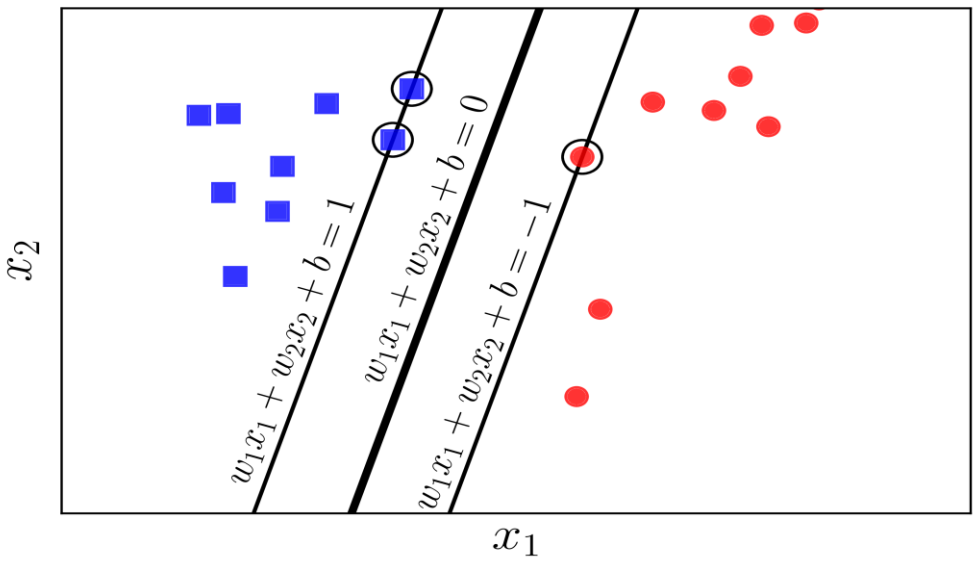
$$margin = \min_i \frac{y_i(\omega_0 + W^T X_i)}{||W||_2}, \forall i \in [1, n]$$

Bài toán tối ưu trong SVM chính là bài toán tìm bộ giá trị tham số (W, ω_0) sao cho lề đạt giá trị lớn nhất:

$$(W, \omega_0) = arg \max_{W,\omega_0} \{ \min_i \frac{y_i(\omega_0 + W^T X_i)}{||W||_2} \}$$

$$= arg \max_{W,\omega_0} \{ \frac{1}{||W||_2} \min_i y_i(\omega_0 + W^T X_i)$$

Giả sử $y_i(\omega_0 + W^T X_i) = 1$ khi đó ta có:



Hình 14 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn.

Như vậy $\forall i \in [1,n]$ ta có: $y_i(\omega_0 + W^T X_i) \geq 1$ Bài toán trở thành bài toán quy hoạch tuyến tính được phát biểu như sau:

$$(W, \omega_0) = arg \max_{(W,\omega_0)} \frac{1}{||W||_2}$$

$$v.đ.k: y_i(\omega_0 + W^T X_i) \geq 1, \forall i \in [1,n]$$

Đánh giá mô hình:

Mô hình Naïve bayes:

Độ chính xác	54.26
Precision (macro)	54.00
Precision (micro)	54.26
Recall (macro)	44.98
Recall (micro)	54.26
Chỉ số - F1 (macro)	43.29
Chỉ số - F1 (micro)	54.26

Mô hình SVM-kernel linear

Độ chính xác	67.35
Precision (macro)	64.99
Precision (micro)	67.35
Recall (macro)	62.86
Recall (micro)	67.35
Chỉ số - F1 (macro)	62.97
Chỉ số - F1 (micro)	67.35

1.2.3 Thuật toán K-Mean trong phân cụm văn bản

Mục đích cuối cùng của thuật toán phân nhóm này là: từ dữ liệu đầu vào và số lượng nhóm chúng ta muốn tìm, chỉ ra center của mỗi nhóm và phân các điểm dữ liệu vào các nhóm tương ứng. (bài toán mỗi điểm dữ liệu chỉ thuộc vào đúng một nhóm-unsupervised).

Thuật toán:

Giả sử có N điểm dữ liệu là X = $[x_1, x_2, \dots, x_N] \in R^{dXN}$ và $K < N$ là số cluster chúng ta muốn phân chia. Chúng ta cần tìm các center $m_1, m_2, \dots, m_K \in R^{dX1}$ và label của mỗi điểm dữ liệu.

Với mỗi điểm dữ liệu x_i đặt $y_i = [y_{i1}, \dots, y_{ij}, \dots, y_{iK}]$ là label vector của nó, trong đó nếu x_i được phân vào cluster j thì $y_{ij} = 1$ và các thành phần còn lại nhận giá trị 0 - (Cách mã hóa label của dữ liệu như thế này được goi là biểu diễn one-hot).

Có thể viết dưới dạng toán học như sau:

$$y_{ij} \in \{0, 1\}, \sum_{j=1}^{K} y_{ij} = 1$$

Hàm mất mát:

Nếu ta coi center m_j là center của mỗi cluster và ước lượng tất cả các điểm được phân vào cluster này bởi m_j, thì một điểm dữ liệu x_i được phân vào cluster j sẽ bị sai số là: $(x_i - m_j)$. Chúng ta mong muốn sai số này có trị tuyệt đối nhỏ nhất nên ta sẽ tìm cách để đại lượng sau đây đạt giá trị nhỏ nhất:

$$\|x_i - m_j\|_2^2$$

Hơn nữa do x_i được phân vào cluster j => $y_{ij} = 1, y_{ik} = 0 \; \forall \; j \; \neq k$. Do đó ta có:

$$y_{ij}\|x_i - m_k\|_2^2 = \sum_{j=1}^{k} y_{ij}\|x_i - m_j\|_2^2$$

Sai số trên toàn bộ dữ liệu:

$$L(Y, M) = \sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij}\|x_i - m_j\|_2^2$$

Trong đó $Y = [y_1, y_2, \dots, y_N]$; $M = [m_1, m_2, \dots, m_k]$ lần lượt là các ma trận được tạo bởi label vector của mỗi điểm dữ liệu và center của mỗi cluster.

Tóm lại chúng ta giải bài toán tối ưu:

$$Y, M = argmin \sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij}\|x_i - m_j\|_2^2$$

$$v\text{đ}k: y_{ij} \in \{0, 1\}, \sum_{j=1}^{K} y_{ij} = 1$$

1.3 Bài toán nhận dạng chữ số viết tay

1.3.1 Mạng neural tích chập

Convolutional Neural Networks (CNNs – Mạng neural tích chập) là một trong những mô hình học sâu tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động.

CNNs được sử dụng nhiều trong các bài toán nhận dạng, phân lớp các đối tượng trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection).

Định nghĩa mạng neural tích chập

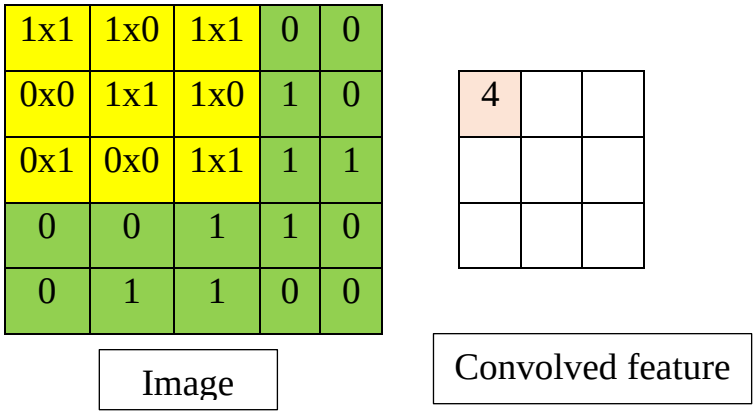
Mô hình mạng neural nhân tạo truyền thẳng ra đời đã được áp dụng rất nhiều trong các bài toán nhận dạng Tuy nhiên mạng neural truyền thẳng không thực hiện tốt lắm với các bài toán dữ liệu nhận dạng hình ảnh. Ví dụ như một hình ảnh có kích thước 32x32 pixel sẽ cho ra vector đặc trưng có 1024 chiều, còn đối với ảnh màu có cùng kích thước sẽ là 3072 chiều. Điều này có nghĩa là ta sẽ cần đến 3072 trọng số nối giữa lớp ẩn kế tiếp, dân đến mô hình quá đồ sộ. điều này càng khó khăn hơn khi thao tác với các ảnh kích thước lớn hơn nữa. Chính vì vậy mạng Convolution Neural Netwworks (CNNs- mạng neural tích chập) ra đời. Mạng neural tích chập (CNNs hoặc ConvNet) là một trong thuật toán học hình ảnh và video phổ biến nhất. Giống như các mạng neural khác, CNNs bao gồm một lớp đầu vào, một lớp đầu ra và nhiều lớp ẩn ở giữa [2].

Các lớp trong mô hình neural tích chập

Tầng trích xuất vector đặc trưng: Khối này bao gồm 3 lớp được sử dụng kết hợp với nhau nhằm mục đích trích xuất đặc trưng, và lựa chọn đặc trưng.

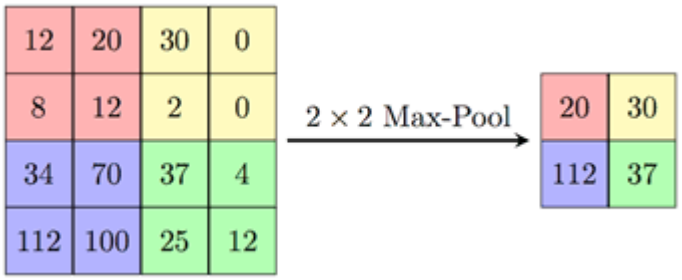
Tích chập: Đây là lớp quan trọng nhất trong cấu trúc của CNNs, đây chính là nơi trích xuất các đặc trưng thể hiện tư tưởng ban đầu của mạng neural tích chập [2]. Thay vì kết nối toàn bộ điểm, lớp này sẽ sử dụng một bộ hạt nhân có kích thước nhỏ hơn so với ảnh (thường là 2x2 cho đến 5x5) áp vào một vùng trong ảnh và tiến hành tích chập giữa bộ hạt nhân này và giá trị điểm ảnh trong

vùng cục bộ đó. Hạt nhân sẽ lần lượt được dịch chuyển theo một giá trị bị trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.



Hình 15 Mô phỏng hoạt động của Convolution

Lớp pooling sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Tuy nhiên, khác với lớp Convlution, lớp pooling không tính tích chập mà tiến hành lấy mẫu (subsampling) [2]. Khi cửa sổ trượt trên ảnh chỉ có một giá trị được xem là giá trị đại diện cho thông tin ản tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong lớp pooling là maxpooling (lấy giá trị lớp nhất), minpooling (lấy giá trị nhỏ nhất) và averagepooling (lấy giá trị trung bình). Xét một ảnh có kích thước 32x32 và lớp pooling sử dụng là filter có kích thước 2x2 và bước trượt stride = 2, phương pháp sử dụng maxpooling. Filter sẽ lần lượt duyệt qua ảnh, với mỗi lần duyệt chỉ có giá trị lớn nhất trong bốn giá trị nằm trong vùng cửa sổ 2x2 của filter được giữ lại và đưa ra đầu ra. Như vậy sau khi qua pooling, ảnh sẽ giảm kích thước xuống 16x16 (mỗi chiều giảm 2 lần).



Hình 16 Max - Pooling

Lớp pooling có vai trò làm giảm kích thước dữ liệu. Với mỗi bức ảnh kích thước lớn qua nhiều lớp pooling sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc cần cho việc nhận dạng (thông tin cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting) [9].

Lớp ReLU – Rectified Linear Unit: về cơ bản tích chập là một phép biến đổi tuyến tính. Nếu tất cả các neural được tổng hợp bởi các phép biến đổi tuyến tính thì một mạng neural đều có thể đưa về dạng một hàm tuyến tính. Khi đó mạng ANN sẽ đưa các bài toán về logistic regression. Do đó tại mỗi neural cần có một hàm truyền dưới dạng phi tuyến [9].

Lớp này sử dụng hàm kích hoạt $f(x) = \max(0, x)$. Nói một cách đơn giản, lớp này có nhiệm vụ chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp tích chập thành giá trị 0. Ý nghĩa của cách cài sặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng đa tầng đa lớp trở nên vô nghĩa. Có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt sigmoid, tanh….

Tầng phân lớp: Sau khi phát hiện kiến trúc của CNNs chuyển sang quá trình phân loại.

Lớp kết nối đầy đủ: lớp này tương tự với lớp trong mạng neural truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào nút trong lớp tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ lớp trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng. Tóm lại, lớp kết nối đầy đủ đóng vai trò như một mô hình phân lớp và tiến hành dựa trên dữ liệu đã được xử lý ở các lớp trước đó.

1.3.2 Xây dựng mô hình bài toán

Input: Sử dụng bộ dữ liệu chữ viết tay MNIST gồm 60000 data training, 10000 data test(Bộ dữ liệu chữ số từ 0 đến 9 viết tay).

Output: phân lớp ảnh gồm 10 lớp 0 => 9.

Đánh giá mô hình:

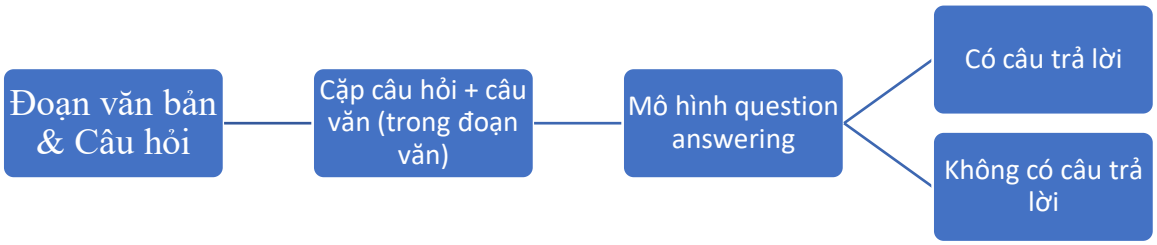
Accuracy	98.75
Precision (micro)	98.75
Precision (macro)	98.71594877896766
Recall (micro)	98.75
Recall (macro)	98.73899988862152
F1-Score (micro)	98.75
F1_Score (macro)	98.72489837616023

CHƯƠNG 2: HỆ THỐNG HỎI ĐÁP (QUESTION ANSWERING)

2.1 Cơ sở lý thuyết

2.1.1 Giới thiệu mô hình hóa bài toán

Bài toán question answering trình bày trong báo cáo này được mô hình hóa theo mô hình dưới đây:



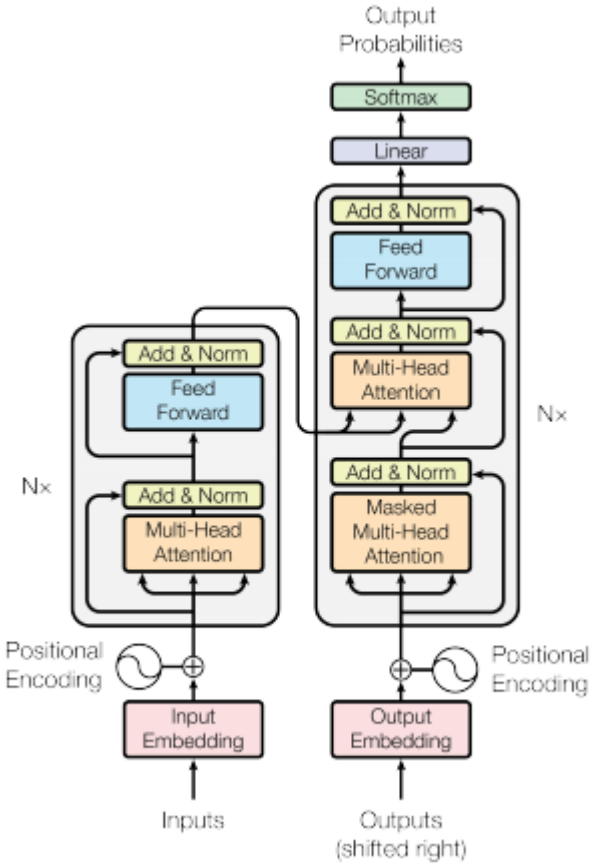
Hình 17 Mô hình hóa bài toán question answering

Đầu vào của bài toán sẽ là một đoạn văn bản tiếng việt bất kỳ và câu hỏi kèm theo, sau đó cho qua mô hình question answering sẽ thu được đầu ra là tính chất của câu văn (thuộc đoạn văn bản) ở một trong hai loại là có chứa câu trả lời cho câu hỏi hoặc không chứa câu trả lời cho câu hỏi.

2.1.2 Mô hình Bert (Bidirectional Encoder Representations from Transformers)

Phương pháp transformer – encoder + decoder trong Bert

Đây là một lớp của mô hình seq2seq (attention layer), bao gồm hai pha là encoder và decoder, và hoàn toàn không có kiến trúc của mạng neural hồi tiếp mà chỉ sử dụng các attention layer xếp chồng lên nhau để embedding các từ trong câu.



Hình 18 Kiến trúc attention layer

Kiến trúc này gồm hai phần encoder bên trái và decoder bên phải.

Encoder: bao gồm 6 layer liên tiếp xếp chồng lên nhau. Mỗi layer bao gồm hai layer con (sub-layer) trong nó. Sub-layer đầu tiên là multi-head self-attention. Layer thứ hai đơn thuần chỉ là các fully-connected feed-forward layer như nhánh bên trái của hình vẽ. Chúng ta sẽ sử dụng một kết nối residual ở mỗi sub-layer ngay sau layer normalization. Đầu ra của mỗi sub-layer là LayerNorm (x+Sublayer(x)) có số chiều là 512.

Decoder: Decoder cũng bao gồm 6 layer xếp chồng lên nhau. Kiến trúc tương tự như các sub-layer ở Encoder ngoại trừ thêm một sub-layer thể hiện phân phối attention ở vị trí đầu tiên. Layer này không gì khác so với multi-head self-attention layer ngoại trừ được điều chỉnh để không đưa các từ trong tương

lai vào attention. Tại bước thứ i của decoder chúng ta chỉ biết được các từ ở vị trí nhỏ hơn i nên việc điều chỉnh đảm bảo attention chỉ áp dụng cho những từ nhỏ hơn vị trí thứ i. Cơ chế residual cũng được áp dụng tương tự như trong Encoder.

Lưu ý là chúng ta luôn có một bước cộng thêm Positional Encoding vào các input của encoder và decoder nhằm đưa thêm yếu tố thời gian vào mô hình làm tăng độ chuẩn xác. Đây chỉ đơn thuần là phép cộng vector mã hóa vị trí của từ trong câu với vector biểu diễn từ. Chúng ta có thể mã hóa dưới dạng [0, 1] vector vị trí hoặc sử dụng hàm sin,cos.

Cấu trúc mô hình Bert

Bert cho nhiệm vụ mask ML:

Khoảng 15 % các token của câu input được thay thế bởi [MASK] token trước khi truyền vào model đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che dấu (non-masked) xung quanh [MASK] và đồng thời là ngữ cảnh cảnh của [MASK] để dự báo giá trị gốc của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít (15%) để tỷ lệ ngữ cảnh chiếm nhiều hơn (85%).

Bản chất của kiến trúc BERT vẫn là một mô hình seq2seq gồm hai phase encoder giúp embedding các từ input và decoder giúp tìm ra phân phối xác suất của các từ ở output. Kiến trúc Transfomer encoder được giữ lại trong tác vụ Masked ML. Sau khi thực hiện self-attention và feed forward ta sẽ thu được các véc tơ embedding ở output là $O_1, O_2, \dots, O_n$.

Để tính toán phân phối xác suất cho từ output, chúng ta thêm một Fully connect layer ngay sau Transformer Encoder. Hàm softmax có tác dụng tính

toán phân phối xác suất. Số lượng units của fully connected layer phải bằng với kích thước của từ điển.

Cuối cùng ta thu được vector nhúng của mỗi một từ tại vị trí MASK sẽ là embedding vector giảm chiều của vector O_i sau khi đi qua fully connected layer như mô tả trên hình vẽ bên phải.

Hàm loss function của BERT sẽ bỏ qua mất mát từ những từ không bị che dấu và chỉ đưa vào mất mát của những từ bị che dấu. Do đó mô hình sẽ hội tụ lâu hơn nhưng đây là đặc tính bù trừ cho sự gia tăng quan hệ về ngữ cảnh của câu. Việc lựa chọn ngẫu nhiên 15% số lượng các từ bị che dấu cũng tạo ra vô số các kịch bản input cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.

Bert cho nhiệm vụ dự đoán câu tiếp theo:

Đây là một bài toán phân loại học có giám sát với hai nhãn (hay còn gọi là phân loại nhị phân). Input đầu vào của mô hình là một cặp câu (pair-sequence) sao cho 50% câu thứ hai được lựa chọn là câu tiếp theo của câu thứ nhất và 50% được lựa chọn một cách ngẫu nhiên từ bộ văn bản mà không có mối liên hệ gì với câu thứ nhất. Nhãn của mô hình sẽ tương ứng với 1 khi cặp câu là liên tiếp hoặc 0 nếu cặp câu không liên tiếp.

Chúng ta cần đánh dấu các vị trí đầu câu thứ nhất bằng token [CLS] và vị trí cuối các câu bằng token [SEP]. Các token này có tác dụng nhận biết các vị trí bắt đầu và kết thúc của từng câu thứ nhất và thứ hai.

Thông tin input được preprocessing trước khi đưa vào mô hình huấn luyện bao gồm:

Ngữ nghĩa của từ (token embeddings): Thông qua các embedding vector cho từng từ. Các vector khởi tạo từ pretrain model.

Ngoài embedding biểu diễn từ của các từ trong câu, mô hình còn embedding thêm một số thông tin:

Loại câu (segment embeddings): Gồm hai véc tơ là E_A nếu từ thuộc câu thứ nhất và E_B nếu từ thuộc câu thứ hai.

Vị trí của từ trong câu (position embedding): là các vector E_1, …, E_N.

Vector input sẽ bằng tổng của cả ba thành phần embedding theo từ, câu và vị trí.

2.1.3 Một số hướng tiếp cận bài toán

Question Answering unsupervised [10]

Bài toán chia làm 3 phase:

- Phase 1: Answer extract:

Tiền xử lý data: loại bỏ các kí tự đặc biệt, các tag html, … chuẩn hóa dữ liệu về dạng viết thường, chia data thành các đoạn văn nhỏ có kích thước cố định, phù hợp với bộ nhớ.

Có hai hướng xử lý:

- ✓ Extract tất cả noun pharse => sử dụng churking algorithm
- ✓ Sử dụng Named Entities => extract answer được đặt tên (Nên sử dụng named entities vì các thực thể được đặt tên giúp dễ sinh câu hỏi và cho độ chính xác question generation cao hơn – khớp với từ để hỏi, tuy nhiên hệ thống câu hỏi tạo ra không phong phú do hạn chế các tag)

Named Entities Recognition:

- ✓ Các thực thể được lưu trữ làm answer, vị trí bắt đầu và kết thúc của nó trong đoạn văn cũng được lưu lại, tag => được dùng để phục vụ cho phase question generation
- ✓

- Phase 2: Question generation

Sử dụng Unsupervised Neural Machine Translation (UNMT)

Yêu cầu: cần tìm một bộ các câu hỏi tiếng việt – coi câu văn (là một ngôn ngữ) => câu hỏi (là một ngôn ngữ khác) => quá trình translate câu văn => câu hỏi và ngược lại.

- Phase 3: Question answering

Sau phase 2: Question generation sẽ thu được một bộ dữ liệu được gán nhãn, thì tại phase 3 việc giải quyết bài toán question answering là bài toán supervised.

Ưu điểm của phương pháp:

- ✓ Phương pháp phù hợp với việc giải quyết bài toán có ít dữ liệu. (Tạo ra một nguồn dữ liệu train vô tận từ dữ liệu thô crawl từ web).
- ✓ Với nguồn dữ liệu lớn được tạo ra sẽ làm dữ liệu train đa dạng về mặt nội dung, câu hỏi làm cho model hoạt động tốt trong dữ liệu thực tế.

Nhược điểm của phương pháp:

- ✓ Các thư viện hỗ trợ tiếng việt trong bài toán name entity recognition (để extract answer) còn hạn chế đặc biệt là extract vị trí của answer trong câu.
- ✓ Cần một lượng lớn dữ liệu câu hỏi phù hợp được với dữ liệu dung extract answer.

- ✓ Kiến trúc bài toán phức tạp

(Em mới chỉ dừng ở mức tìm hiểu chưa thể thực nghiệm)

Question Answering sử dụng pretrain model Roberta For Sequence Classification

RobertaForSequenceClassification (là pretrain model dùng phân lớp được cung cấp từ phoBert VinAI)

Có cấu trúc cơ bản bert base (gồm 12layer attention).

Khác biệt lớn nhất là đầu ra dùng để phân lớp dữ liệu (bao gồm: linear layer + dropout layer + linear layer + sofmax layer.

Sử dụng tokenizer phoBert để extract feature trước khi đưa vào pretrain model để phân lớp.

Sau đó qua pretrain model => Đầu ra xác định lớp của input

Ưu điểm:

- ✓ Finetune model đơn giản, không cần chỉnh sửa (do model được xây dựng sẵn cho nhiệm vụ phân loại).
- ✓ Chỉ cần xử lý dữ liệu, extract feature cần thiết cho model => predict output.

Nhược điểm:

- ✓ Do model được xây dựng cho bài toán phân loại (nhưng không có phần phân biệt câu trong input đầu vào => nên việc sử dụng cho bài toán Q&A cần phân biệt question và answer cho kết quả không tốt.

- ✓ Hiệu suất hoạt động của model lâu, tốn kém tài nguyên.

  (lấy max_seq_lengh = 100 => chạy gpu (colab) => tràn RAM. Chạy trên cpu (colab) tràn memory)

- ✓ Test thử max_seq_length = 50 model hoạt động tương đối tốt cho acc: 78.84, F1_score = 79.09, recall_score = 78.84. Tuy nhiên khi test thực tế kết quả rất tồi (ngoài ra khi visual dữ liệu độ dài answer và question tương đối dài việc max_len=50 => mất mát lượng lớn thông tin.

Question Answering (Sử dụng pretrain model bert base multilingual uncased) [11]

Ưu điểm:

- ✓ Pretrain model bert base multilingual uncased hỗ trợ trên nhiều ngôn ngữ (102 ngôn ngữ - có cả tiếng Việt). Được đào tạo trên nguồn data khổng lồ => có khả năng tốt hơn model mình train nếu dữ liệu không đủ nhiều và tốt.
- ✓ Việc finetune model tương đối đơn giản => cho kết quả khá khả thi.

Nhược điểm:

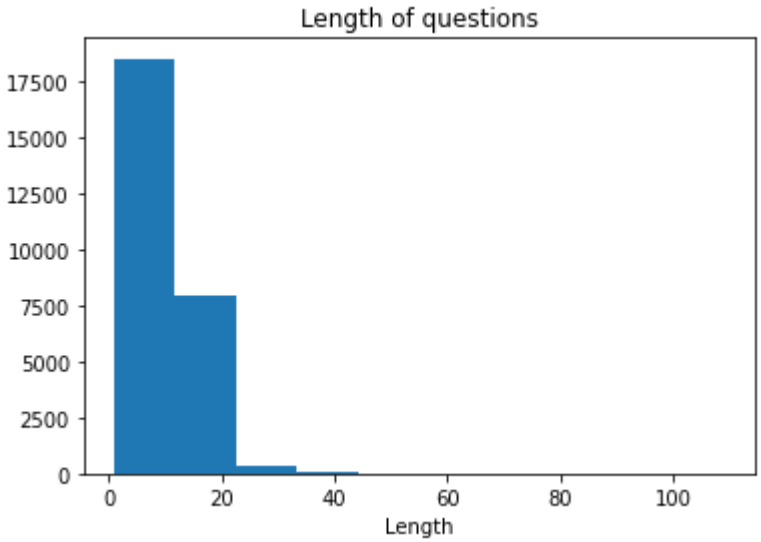
- ✓ Tuy nhiên model này phục vụ cho bài toán answer selection => việc loại bỏ một số parameter sẽ làm giảm độ chính xác của mô hình.

2.2 Question Answering (Sử dụng pretrain model bert base multilingual uncased)

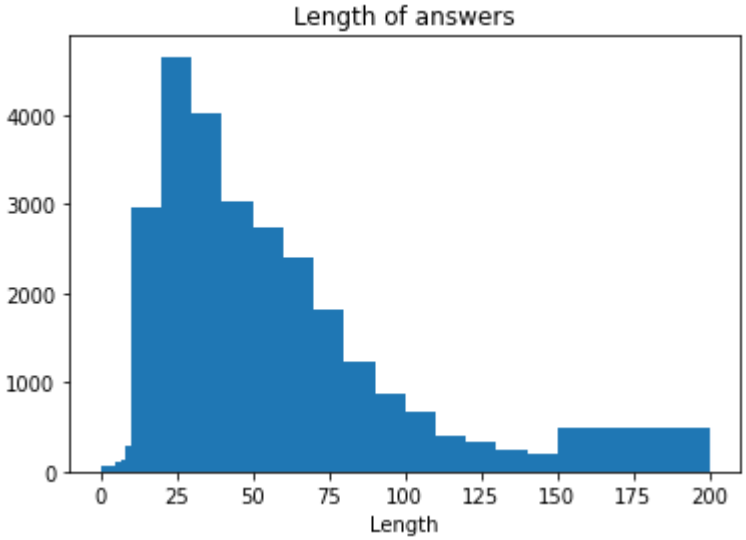
2.2.1 Mô tả dữ liệu

Dữ liệu training: Sử dụng bộ dữ liệu của cuộc thi Zalo AI challenge. Bao gồm: 26930 dữ liệu. Có bổ sung dataset WikiQA (được translate sang tiếng việt).

=> Tổng bộ dữ liệu bao gồm: 47287 dữ liệu train, 6116 dữ liệu val, 2733 dữ liệu test (Dữ liệu khá mất cân bằng (khoảng 27% nhãn True, 73% nhãn False)).



Hình 19 Biểu đồ phân bố độ dài câu hỏi



Hình 20 Biểu đồ phân bố câu trả lời

Khai thác dữ liệu

Answer_text: chỉ là một câu được trích ra trong đoạn văn.

Số lượng answer_text nhãn False: chủ yếu đều là các câu không mang bất cứ thông tin nào của câu hỏi. Tuy nhiên, có tương đối câu gây nhiễu => predict sai, ảnh hưởng lớn đến model. Ví dụ:

- ✓ Các câu có chứa nhiều keyword của câu hỏi nhưng lại không chứa answer.
- ✓ Các câu có mang thông tin câu trả lời nhưng lại dùng đại từ nhân xưng, … thay thế answer cần extract.

Một số sai lầm trong quá trình xây dựng và đào tạo mô hình:

Trong lúc split dữ liệu để train đã gặp phải sai lầm nghiêm trọng (do câu hỏi sẽ được lặp lại nên dung split random nhưng trong tập test hoặc val đều chứa câu hỏi của train => đánh giá test cao vượt trội so với train

Dữ liệu train đã được làm sạch, và hầu như không có từ viết tắt, trong khi dữ liệu thực tế thì chưa được làm sạch, khá nhiều từ viết tắt => ảnh hưởng đến độ chính xác của model.

Thiết kế dữ liệu huấn luyện:

Đối với dữ liệu train đã được làm sạch và xử lý sẵn (không cần tiền xử lý.

Đối với dữ liệu thực tế sử dụng mô hình:

Đoạn văn sẽ cắt thành các câu (loại bỏ các ký tự đặc biệt, các thẻ html …), và sau đó ghép với câu hỏi và xử lý tương tự dữ liệu train trước khi đưa vào dự đoán (Nhược điểm với những đoạn văn dài việc cắt câu và dự đoán sẽ tốn thời gian. Một số câu có sử dụng đại từ nhân xưng thay thế cho chủ thể của câu trước đó do đó những câu như thế sẽ không cho kết quả trả lời).

2.2.2 Kiến trúc mô hình bài toán

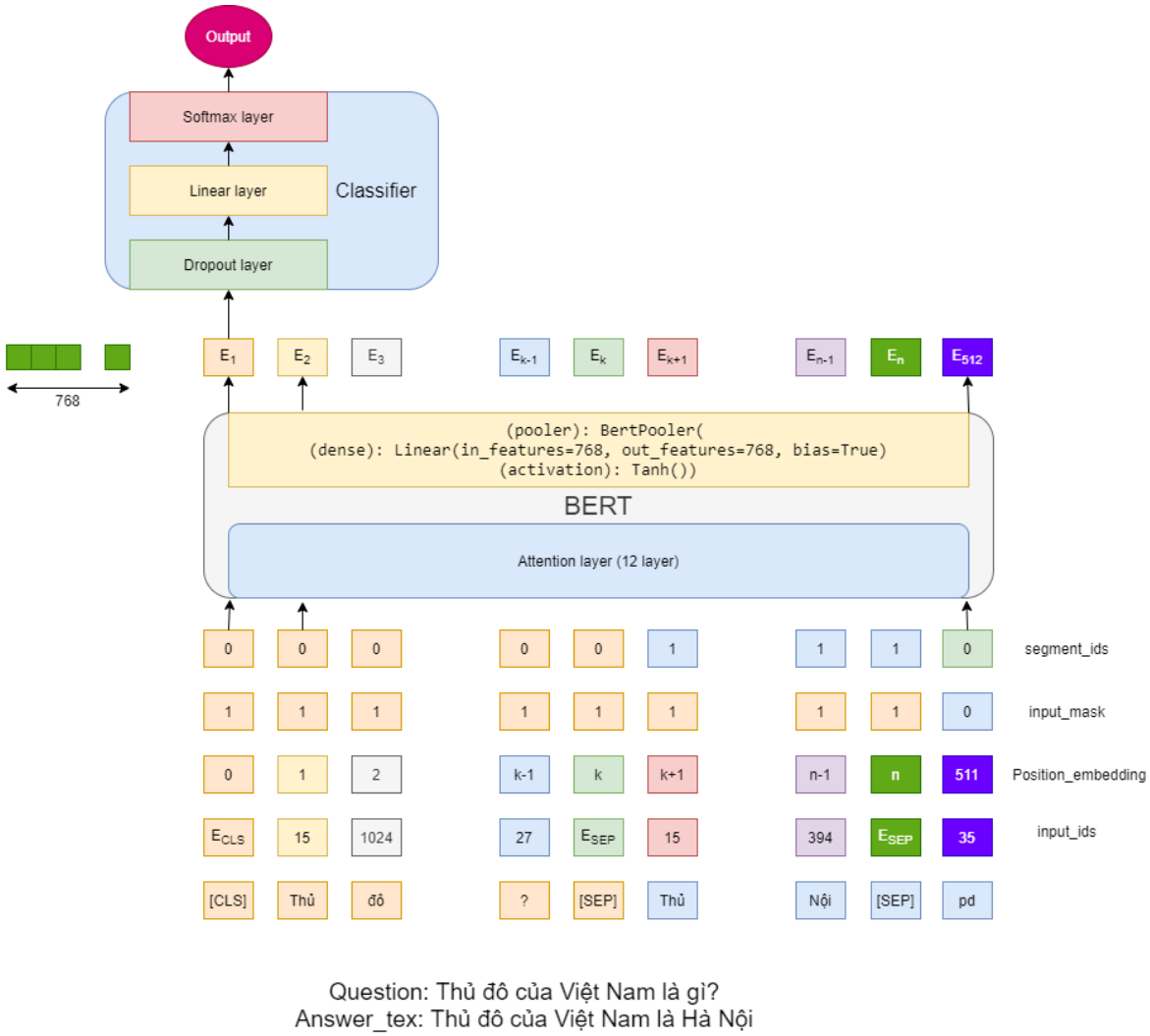
Trích xuất vector đặc trưng:

- ✓ Convert token: [CLS] Question [SEP] Answer_text [SEP]
- ✓ input model: convert token => feature sử dụng trong model:

Khởi tạo một số tham số trước khi xử lý dữ liệu:

- ✓ max_seq_length = 512, max_query_lenght=64
- ✓ input_ids: mã hóa từ dựa trên thứ tự của từ trong bộ từ điển
- ✓ input_mask: phân biệt các token tham gia đào tạo (1) với padding được thêm vào (0)
- ✓ token_type_ids: dung để phân biệt 2 câu: question: 1, answer_text: 0
- ✓ Label: one_hot encoding

Kiến trúc mô hình như sau:



Hình 21 Kiến trúc mô hình question answering sử dụng pretrain model Bert

Đánh giá mô hình:

Epoch_step	Accuracy	F1-score	Recall	Precision
Epoch0_step90 (lr=4.5e-05)	80.00	61.99	61.34	62.86
	87.17	72.02	68.14	82.04
Epoch0_step95 (lr=4.75e-05)	80.29	62.30	61.52	63.41
	83.06	74.04	77.56	72.02
Epoch0_step105 (lr=5.25e-05)	80.45	62.94	62.11	64.14
	79.98	71.95	78.35	69.82
Epoch0_step125 (lr =1e-04)	81.11	63.71	62.63	65.4
	86.07	76.14	76.49	75.82
Epoch0_step210 (lr=8e-05	82.18	66.84	65.43	69.08
	84.62	73.15	73.10	73.21
Epoch0_290 (lr=4e-05)	82.27	67.13	66.01	70.21
	84.65	77.30	77.04	77.73

Link github: https://github.com/hanghust/internt_project_demo

KẾT LUẬN

Như vậy sau thời gian thực tập tại công ty, mặc dù còn gặp nhiều khó khăn nhưng em đã tìm hiểu, training và tích lũy những kiến thức cơ bản và thiết thực về lĩnh vực máy học. Đồng thời, đây là cơ hội vô cùng quý giá và hữu ích để em làm quen với văn hóa doanh nghiệp, kỹ năng giao tiếp, ứng xử tại doanh nghiệp và các kỹ năng mềm, kỹ năng chuyên môn khác.

Một số bài học kinh nghiệm mà em rút ra được trong quá trình thực tập tại công ty:

- ✓ Trang bị các kiến thức lí thuyết trên trường lớp là vô cùng cần thiết giúp cho việc tiếp cận với những bài toán thực tế dễ dàng hơn.
- ✓ Khi làm việc trong môi trường doanh nghiệp chuyên nghiệp, thì kiến thức và kỹ năng chuyên môn là chưa đủ, chúng ta cần phải học tập, trang bị và rèn luyện các kĩ năng mềm khác như kỹ năng quản lý thời gian, kỹ năng làm việc nhóm, kỹ năng báo cáo, thuyết trình…, đặc biệt là kỹ năng giao tiếp ứng xử với mọi người và thái độ khi làm việc, cần phải nghiêm túc hoàn thành nhiệm vụ của mình trước thời hạn được giao.
- ✓ Nên học hỏi những người xung quanh để có thể rút ra những kinh nghiệm cho chính bản thân mình.
- ✓ Cần chủ động, mạnh dạn trao đổi ý kiến, hỏi anh chị hướng dẫn.

Cuối cùng em xin gửi lời cảm ơn các thầy cô trong Viện Toán Ứng dụng và Tin học trường Đại học Bách Khoa Hà Nội và toàn thể các anh chị nhân viên trong Công ty iCOMM Media & Tech, Jsc đã tạo điều kiện và giúp đỡ em trong quá trình thực tập.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] Prabhu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," 4 Mar 2018. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[2] A. C. M. a. S. Guido, Introduction to Machine Learning with Python, FILL IN PRODUCTION EDI-, June 2016.

[3] T. T. T. Minh Quang Nhat Pham, "A Lightweight Ensemble Method for Sentiment," FPT Technology Research Institute (FTRI), 2016.

[4] E. R. M. Ralf C. Staudemeye, "– Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," September 23, 2019.

[5] T. C. a. N. Bayes, Stanford.

[6] N. R. Kavya Suppala, "Sentiment Analysis Using Naïve Bayes Classifier," June, 2019.

[7] C. M. Bishop, Pattern recognition and Machine Learning, Springer (2006).

[8] R. O. P. E. H. a. D. G. S. J. W. &. S. Duda, "Pattern classification," 2012.

[9] P. Punyawiwat, "Interns Explain CNN," 8 Feb 2018. [Online]. Available: https://blog.datawow.io/interns-explain-cnn-8a669d053f8b.

[10] L. D. S. R. Patrick Lewis, "Unsupervised Question Answering by Cloze Translation," Association for Computational Linguistics, july-2019.

[11] Y.-H. C. •. Y.-C. Fan, "A Recurrent BERT-based Model for Question Generation," Association for Computational Linguistics, November-2019.

Các slide bài giảng môn “Học máy’ của thầy Ngô Văn Linh, Viện Công nghệ Thông tin và Truyền thông Đại học Bách Khoa Hà Nội, 2018