

**TỔNG CÔNG TY VIỄN THÔNG VIETTEL**

**KHỐI CÔNG NGHỆ THÔNG TIN**

**PYC THỬ NGHIỆM ĐÁNH GIÁ CÁC MÔ HÌNH MACHINE LEARNING VÀ DEEP LEARNING**

**TÀI LIỆU MÔ TẢ YÊU CẦU**

**Mã hiệu dự án:** **MyViettel\_0009**

**Mã hiệu tài liệu: MyViettel\_0009**

**<Địa điểm, Thời gian>**

**BẢNG GHI NHẬN TIẾN ĐỘ**

\*A - Tạo mới, M - Sửa đổi, D - Xóa bỏ

| **Ngày**  **bàn giao** | **Yêu cầu** | **A\***  **M, D** | **Trạng thái** | **Đầu mối KH** | **Mô tả** | **Ghi chú** |
| --- | --- | --- | --- | --- | --- | --- |
| 20/4/2023 | Đánh giá các nhóm mô hình Machine learning và Deep learning | A | Hoàn thành | PTC, PKH | * Mô tả chi tiết, đánh giá performance mô hình |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**MỤC LỤC**

[**I. Mở đầu 7**](#_heading=h.2et92p0)

[1.1. Khảo sát nghiệp vụ 7](#_heading=h.tyjcwt)

[**II. Đề xuất và xây dựng nhóm mô hình Deep Learning 7**](#_heading=h.bqtj5kzy11c)

[2.1. Mô hình sử dụng Contrastive Learning 7](#_heading=h.qxvhd1spdg7n)

[2.1.1. Tổng quan về Contrastive Learning 7](#_heading=h.p7tqm6tc132g)

[2.1.2. Những vấn đề liên quan 8](#_heading=h.e4cugbvpefei)

[2.1.2.1 Data augmentation 8](#_heading=h.t9wuneshaw23)

[2.1.2.2 Encoder 9](#_heading=h.3978qos6hzpr)

[2.1.2.3 Projection Head 9](#_heading=h.68gb8ikf33mc)

[2.1.2.4 Noise Contrastive Estimation (NCE) 10](#_heading=h.okjtgkuznqk)

[2.1.2.5 Pretext Task 10](#_heading=h.nn7vptknu3de)

[2.1.2.6 DownstreamTask 11](#_heading=h.2azn94lp4bcv)

[2.1.3. Kiến trúc mô hình sử dụng Contrastive Learning: DouRec 11](#_heading=h.ogn8mtv6p937)

[2.1.3.1 Embedding Layer 11](#_heading=h.mzwnsxz6dlbx)

[2.1.3.2 Self-attention Module 12](#_heading=h.zcs0nwtsgdm)

[2.1.3.3 Recommended learning 12](#_heading=h.hsiv06729foy)

[2.1.3.4 Contrastive regularization 12](#_heading=h.hw4j8kxfzdw4)

[2.2. Mô hình BERT4Rec 14](#_heading=h.9qqrlp3yqeii)

[2.2.1. Giới thiệu chung 14](#_heading=h.tvgx5yqr78zu)

[2.2.2. Những vấn đề liên quan 17](#_heading=h.2lo17wggf4m4)

[2.2.2.1. Hệ thống recommendation tổng quan 17](#_heading=h.2kjxh8mooxeo)

[2.2.2.2. Hệ thống recommendation tuần tự 18](#_heading=h.ltrihv52fj2k)

[2.2.2.3. Cơ chế Attention 18](#_heading=h.dks70s9f2tgi)

[2.2.3. Kiến trúc mô hình BERT4Rec 19](#_heading=h.5znbrsccnmwi)

[2.2.3.1 Vấn đề Statement 19](#_heading=h.w11nrorm0c2j)

[2.2.3.2. Kiến trúc mô hình tổng quan 19](#_heading=h.dcg8paxpeiai)

[2.2.3.3. Transformer Layer 21](#_heading=h.dxf513281c9p)

[2.2.3.4. Embedding Layer 23](#_heading=h.fugthwjvzh8v)

[2.2.3.5. Output Layer 24](#_heading=h.tuua9kofwt3e)

[2.2.3.6 Quá trình huấn luyện 24](#_heading=h.622of2r8o2lz)

[2.3. Mô hình LSTM 25](#_heading=h.kpdmvf41nr95)

[**III. Đề xuất và xây dựng nhóm mô hình Machine Learning 26**](#_heading=h.qjm75kolj8t0)

[3.1. Mô hình Linear Regression 26](#_heading=h.yjo9tj40imt)

[3.1.1. Tổng quan về mô hình Linear Regression 26](#_heading=h.fk8wltpjxtp8)

[3.1.2. Các vấn đề liên quan 28](#_heading=h.a3of41ikxlze)

[3.1.2.1 Loss function (Hàm mất mát) 28](#_heading=h.twuwwxktg89q)

[3.1.2.2 Gradient Descent trong Linear Regression 28](#_heading=h.74ddnlfh0uao)

[3.1.2.3 Phương thức đánh giá mô hình Linear Regression 30](#_heading=h.tytrrkffgykt)

[3.2. Mô hình Logistic Regression 32](#_heading=h.i69f96l7td9)

[3.2.1. Tổng quan về mô hình Logistic Regression 32](#_heading=h.6p9iwlr75rkv)

[3.2.2. Các vấn đề liên quan 33](#_heading=h.967mu7ncc03c)

[3.2.2.1 Hàm Sigmoid 33](#_heading=h.r9vz8mdslpnb)

[3.2.2.2 Hàm mất mát (Loss function) 34](#_heading=h.10z8y49iqriz)

[3.2.2.3 Phương pháp tối ưu (Optimization) 36](#_heading=h.ai5bbin6b27s)

[3.2. Mô hình SVM 37](#_heading=h.we31sdx07fc4)

[3.2.1. Tổng quan về mô hình SVM 37](#_heading=h.vk794w2sxez)

[3.2.2. Những vấn đề liên quan 40](#_heading=h.wrbwwocuyq8j)

[**IV. Kiểm thử 42**](#_heading=h.yj4magc0dfyg)

[3.1. Kết quả test perfomance 42](#_heading=h.147n2zr)

[3.2. Đánh giá và kết luận 42](#_heading=h.6i6dd6silyzr)

[**Tài liệu tham khảo 42**](#_heading=h.81oh63uongl2)

**TRANG KÝ**

Người lập: <Ngày>

<Chức danh>

Người xem xét: <Ngày>

<Chức danh>

Người xem xét: <Ngày>

<Chức danh>

Người phê duyệt: <Ngày>

<Chức danh>

# I.   Mở đầu

## 1.1. Khảo sát nghiệp vụ

# II.  Đề xuất và xây dựng nhóm mô hình Deep Learning

## 2.1. Mô hình sử dụng Contrastive Learning

### 2.1.1. Tổng quan về Contrastive Learning

Ý tưởng chính của Contrastive learning là tìm ra các cặp đặc trưng của dữ liệu có tính tương đồng - tương phản nhau trong bộ dataset. Từ đó, với những cặp dữ liệu mang tính tương đồng ta có thể "kéo" chúng lại gần để học được những đặc trưng cấp cao hơn của nhau, và ngược lại với những cặp những liệu tương phản sẽ bị "đẩy" ra xa. Để làm được điều này, ta sẽ cần sử dụng các similarity metric để tính toán khoảng cách giữa các embedding vector biểu diễn các điểm dữ liệu với nhau. Ví dụ, ta đã có 1 điểm dữ liệu gốc gọi là anchor, sau đó có thể dùng thêm các kỹ thuật augmentation khác nhau để có thêm 1 biến thể từ anchor gốc gọi là positive sample, và phần còn lại của batch / dataset sẽ được coi là negative sample. Sau đó model sẽ được train để có thể phân biệt được positive sample với negative sample từ 1 cụm dữ liệu.

Lấy một ví dụ đơn giản như sau. Con người chúng ta có thể dễ dàng phân biệt được các vật thể khác nhau từ khi còn bé ( thậm chí là sơ sinh), và những vật thể đấy sau này sẽ được người lớn "gán nhãn". Vậy tại sao chúng ta lại có thể phân biệt được các vật thể đấy từ lần nhìn đầu tiên ? Bởi vì não bộ của chúng ta có thể nhận biết được những đặc điểm, đặc trưng khác biệt nhất của một vật thể hay còn gọi là đặc trưng bậc cao, và đem so sánh những đặc trưng đấy với "dữ liệu" có sẵn trong não bộ để có thể phân biệt được những vật thể giống - khác loại ( loài ). Chỉ cần nhận ra những điểm tương đồng và khác biệt giữa vật khác nhau, bộ não có thể học được các đặc điểm cấp cao của các đối tượng trong thế giới của chúng ta. Ví dụ, chúng ta có thể nhận ra một cách vô thức rằng hai con mèo ở ảnh trên có đôi tai nhọn, trong khi con chó có đôi tai cụp xuống. Hoặc chúng ta có thể đối chiếu chiếc mũi nhô ra của chó với mặt phẳng của mèo. Về cơ bản thì contrastive learning cũng cho phép model của chúng ta làm điều tương tự, mục tiêu của phương pháp là đối chiếu sự tương phản giữa embedding của các phiên bản biến đổi của cùng một sample để học các đặc trưng bất biến của nó trong khi vẫn phân biệt được embedding của các sample khác.

Các phương pháp supervised learning truyền thống phụ thuộc rất nhiều vào lượng dữ liệu đã được gán nhãn sẵn trong khi lượng data chưa được gán nhãn là vô cùng khổng lồ. Mặc dù số dataset đã có nhãn là rất lớn nhưng với sự gia tăng mạnh mẽ của nhu cầu về lượng dữ liệu cũng như các bài toán mới thì số dataset đấy là hoàn toàn không đủ. Nhất là với những bài toán cần gán nhãn chính xác đến từng pixel như semantic segmentation thì đây là việc vô cùng tốn công sức và thời gian. Và từ đấy các phương pháp học self/semi - supervised learning lên ngôi. Với các kỹ thuật self-supervised learning thì chúng ta có thể train để model học rất tốt từ data chưa được gán nhãn và self - supervised learning được áp dụng khá phổ biến với 2 hướng đi : GANs và Contrastive learning. Trong vài năm qua, các phương pháp self - supervised learning tốt nhất đã dần dần chuyển từ pretext task learning như rotation, colorization và jigsaw puzzling sang contrastive learning và xu hướng này đang ngày càng được củng cố nhờ việc có nhiều hơn những nghiên cứu mang tính đột phá trong computer vision với contrastive learning.

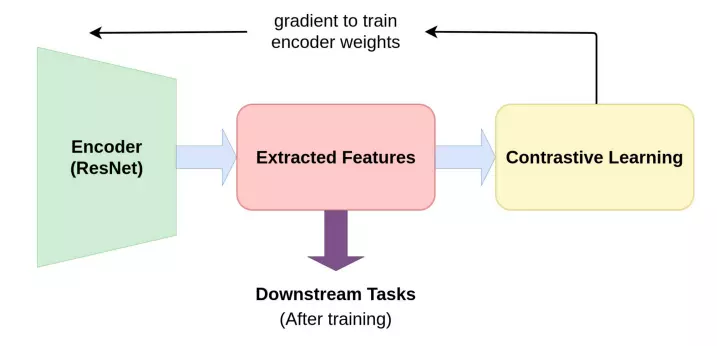
### 2.1.2. **Những vấn đề liên quan**

#### 2.1.2.1 Data augmentation

Data augmentation là 1 phần khá quan trọng trong các mô hình contrastive learning và gần như là không thể thiếu được. Nếu data augmentation không đủ phức tạp thì gradient sẽ không đủ tốt để model có thể học các đặc trưng trong dữ liệu và ngược lại. Trong những nghiên cứu trước về contrastive learning, các tác giả chủ yếu sử dụng tuần tự 3 loại augment : random crop, random color distortions, và random Gaussian blur. Và nhận ra rằng random crop và color distortion là khá quan trọng để model đạt được kết quả tốt. Nhưng nếu chỉ sử dụng riêng rẽ 1 loại biến đổi thì sẽ không cho ra kết quả vượt trội, chỉ khi kết hợp 2 loại biến đổi với nhau thì kết quả mới đạt tới " state-of-the-art ".

#### 2.1.2.2 Encoder

Thông thường, encoder là 1 mạng CNN có nhiệm vụ map các đầu vào là ảnh thành các embedding vector làm đầu vào cho contrastive loss. Nếu mạng encoder trích xuất đặc trưng không đủ tốt thì model sẽ rất khó để học đươc cách phân biệt các đặc điểm có trong 1 điểm dữ liệu. Đa số mạng encoder được sử dụng hiện nay là Resnet và biến thể của nó, trong số đó thì có lẽ Resnet-50 là biến thể được các nhà nghiên cứu sử dụng nhiều nhất vì tính cân bằng giữa kích thước và hiệu suất của mô hình.



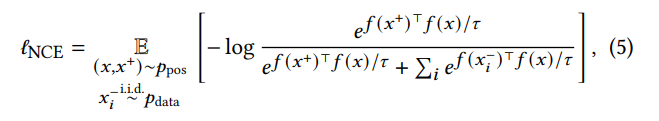
Để model encoder học được các đặc điểm của vật thể với hướng đi của self-supervised thì phải trải qua pretext task ( sẽ được nói ở phần sau ) sau khi được huấn luyện qua với pretext task thì model sẽ có khả năng trích xuất các đặc trưng tốt hơn, đồng thời có thể kiêm thêm downstream task. Một số model thuộc lĩnh vực unsupervised learning thậm chí còn có thể "outperform" các model SoTA của supervised learning, tiêu biểu như MoCo của Facebook hay SimCLR của Google. Điều này có nghĩa là các nhà nghiên cứu đã dần thu hẹp được khoảng cách giữa unsupervised và supervised representation learning trong một số task computer vision.

#### 2.1.2.3 Projection Head

Sau khi data đi qua khối encoder thì sẽ được tính toán với projection head được cấu thành từ các fully-connected layer để cho ra output là 1 embedding vector biểu diễn cho ảnh ở đầu vào. Mạng này có tác dụng khuếch đại các đặc trưng bất biến của dữ liệu từ đó tối đa hóa khả năng phân biệt các phép biến đổi từ cùng 1 bức ảnh của mạng.

#### 2.1.2.4 Noise Contrastive Estimation (NCE)

Contrastive Learning là một cơ chế huấn luyện “kéo” các cặp dữ liệu tương đồng lại gần và đẩy các cặp dữ liệu tương phản ra xa. Cụ thể, hàm mất mát Noise Contrastive Estimation (NCE) thường được áp dụng để huấn luyện encoder (sẽ trình bày ở phần sau) tạm gọi là 𝑓 :



Trong đó xvà x+  là một cặp dữ liệu tương đồng được lấy từ ppos, mặt khác xvà xi-  là một cặp dữ liệu tương phản trong đó xi- được lấy ngẫu nhiên trong pdata . 𝜏 là tham số điều chỉnh.

#### 2.1.2.5 Pretext Task

Pretext task chính là self-supervised task với nhiệm vụ huấn luyện để model học được các đặc trưng với dữ liệu chưa được đánh nhãn, với các phương pháp thông thường thì các điểm dữ liệu sẽ được gán nhãn giả bằng cách: biến đổi ảnh gốc dựa trên các phép data augmentation (augmented data) làm input cho model và output sẽ là ảnh gốc đấy. Tức là chúng ta sẽ để model học được cách khôi phục ảnh gốc dựa trên các bức ảnh đã bị biến đổi. Mục tiêu của pretext task thông thường khác pretext task của contrastive learning - contrastive prediction task ở chỗ pretext task sẽ cố gắng khôi phục lại ảnh cũ từ ảnh đã biến đổi, còn contrastive prediction task sẽ cố gắng học những đặc trưng bất biến của ảnh gốc từ ảnh đã biến đổi trong khi vẫn phân biệt được với các ảnh negative khác trong batch. Model đã được huấn luyện qua pretext task có thể được sử dụng ở các nhiệm vụ khác như fine-tune với các bộ dataset đã gán nhán để thực hiện các downstream task. Có 4 loại pretext task thường được dùng đó là : color transformation, geometric transformation, context-based tasks, và cross-modal-based tasks.

#### 2.1.2.6 DownstreamTask

Downstream task nói nôm na chính là việc chúng ta tận dụng các kiến thức model đã được học qua pretext task để sử dụng chúng vào một mục đích cụ thể nào đó. Model đã được pretrain ở pretext task chứa những trọng số giàu thông tin và hoàn toàn có thể được sử dụng với những bài toán thích hợp khác thông qua kỹ thuật fine-tune model hay tổng thể được gọi là transfer learning. Kết quả của transfer learning với những task có level cao hơn kia sẽ miêu tả khả năng tổng quát hóa của các đặc trưng đã được học. Một số downstream task thường được biết đến là classification, detection, segmentation, future prediction. Việc sử dụng một hay nhiều trong các task đã được liệt kê cũng có thể coi là đang kiểm thử và đánh giá hiệu quả của model. Để đánh giá các đặc trưng đã được học của hướng self-supervised cho các downstream task, người ta thường sử dụng các phương pháp như kernel visualization, feature map visualization, và nearest-neighbor-based. Những phương pháp này cũng hỗ trợ việc phân tích hiệu quả của pretext task trong training model

### 2.1.3. Kiến trúc mô hình sử dụng Contrastive Learning: DouRec

Trong hệ thống gợi ý tuần tự, ý tưởng chính là tổng hợp các tương tác lịch sử để lập hồ sơ tùy chọn của người dùng. Mô-đun encoding của DuoRec là Transformer. Để tận dụng khả năng mã hóa của Transformer, các mục trước tiên được chuyển đổi thành các embeddings. Sau đó, mô-đun multi-head self-attention được áp dụng để tính toán biểu diễn người dùng.

#### 2.1.3.1 Embedding Layer

Trong DuoRec, có một ma trận embedding 𝑽 ∈ R|V |×𝑑, trong đó 𝑑 là kích thước của phần embedding. Đối với dãy đầu vào 𝑠 = [𝑣1, 𝑣2, . . . , 𝑣𝑡], các biểu diễn embedding là 𝒔 = [𝒗1, 𝒗2, . . . , 𝒗𝑡], trong đó 𝒗∗ là vectơ được embedding.

Để duy trì thứ tự thời gian của chuỗi, một ma trận mã hóa vị trí 𝑷 ∈ R𝑁 ×𝑑 được xây dựng, trong đó 𝑁 biểu thị độ dài tối đa của tất cả các chuỗi. Thông thường, phần embedding và mã hóa vị trí được thêm vào làm vectơ đầu vào cho tương tác tại bước thời gian 𝑡 của Transformer:

𝒉0𝑡 = 𝒗𝑡 + 𝒑𝑡

trong đó, 𝒉0𝑡 ∈ Rd là vectơ đầu vào đầy đủ của tương tác tại 𝑡 và 𝒑𝑡 là mã hóa vị trí của time step 𝑡.

#### 2.1.3.2 Self-attention Module

Sau khi có được các chuỗi đầu vào,Transformer được áp dụng để tính toán các biểu diễn được cập nhật của từng phần theo cơ chế multi-head attention. Giả sử 𝑯0 = [𝒉00 , 𝒉01, . . . ,𝒉0t] i là biểu diễn ẩn của trình tự vừa là đầu vào của bộ 𝐿-layer multi-head Transformer encoder (Trm), quy trình mã hóa của trình tự có thể được định nghĩa là:



trong đó vectơ hidden cuối cùng 𝒉Lt trong tập 𝑯L = [𝒉L0 , 𝒉L1, . . . ,𝒉Lt] được chọn làm đại diện cho chuỗi người dùng này.

#### 2.1.3.3 Recommended learning

Nhiệm vụ dự đoán mục tiếp theo được đóng khung dưới dạng nhiệm vụ phân loại trên toàn bộ tập hợp mục. Với biểu diễn trình tự 𝒉 và ma trận embedding 𝑽, điểm dự đoán được tính như sau:



trong đó 𝒚ˆ ∈ R|V|. Với chỉ mục của ground truth được chuyển đổi thành vectơ một chiều 𝒚, hàm mất mát cross-entropy được tính như sau:



#### 2.1.3.4 Contrastive regularization

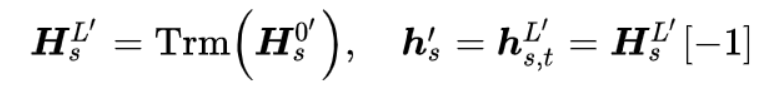
**Unsupervised Augmentation**

Contrastive Augmentation không giám sát trong DuoRec nhằm mục đích cung cấp khả năng tăng cường có ý nghĩa về mặt ngữ nghĩa cho các chuỗi riêng lẻ. Trong quá trình tính toán các vectơ trình tự, có các mô-đun Dropout trong cả lớp embedding và Transformer encoder. Chuyển tiếp chuỗi đầu vào hai lần với các mặt nạ loại bỏ khác nhau sẽ tạo ra hai vectơ khác nhau giống nhau về mặt ngữ nghĩa nhưng có các tính năng khác nhau. Do đó, trong phần tăng cường không giám sát của chuỗi đầu vào trong bài viết này, các mặt nạ loại bỏ khác nhau được chọn và đầu vào embedding của bộ mã hóa được vận hành thêm để thu được 𝒉0’t , sau đó được đưa vào bộ mã hóa, thông qua mã mặt nạ Dropout khác nhau để thu được các đầu ra khác nhau, công thức như sau, sau khi mã hóa, biểu diễn cuối cùng h ' cũng được sử dụng để biểu diễn biểu diễn chuỗi tổng thể:



**Supervised Positive Sampling**

Contrastive Augmentation có giám sát trong DuoRec nhằm mục đích kết hợp thông tin ngữ nghĩa giữa các trình tự tương tự về mặt ngữ nghĩa vào quá trình contrastive regularization . Nếu chỉ áp dụng phương pháp học tương phản không giám sát, các mẫu ban đầu giống nhau về mặt ngữ nghĩa sẽ được phân loại là mẫu tương phản. Do đó, điều tối quan trọng là xác định mẫu nào giống nhau về mặt ngữ nghĩa. **Semantic Similarity:** nếu hai chuỗi đại diện cho cùng một sở thích của người dùng, thì điều tự nhiên là suy ra rằng hai chuỗi chứa cùng một ngữ nghĩa. Do đó, đối với các chuỗi khác nhau 𝑠i = [𝑣i,1, 𝑣i,2, . . . , 𝑣i,𝑡i] và 𝑠j = [𝑣j,1, 𝑣j,2, . . . , 𝑣j,𝑡j] nếu các mục tiêu dự đoán của 𝑠i và 𝑠j, tức là 𝑣i,𝑡i+1 và 𝑣j,𝑡j+1, là cùng một mục, thì 𝑠𝑖 và 𝑠𝑗 được coi là giống nhau về mặt ngữ nghĩa trong DuoRec. **Positive Sampling:** đối với một chuỗi đầu vào s, những chuỗi có cùng target tồn tại trong tập dữ liệu. Một trình tự tương tự về mặt ngữ nghĩa ss được chọn ngẫu nhiên từ các trình tự này . Do đó, đặc tính của nó đã thu được thông qua các bước trên.

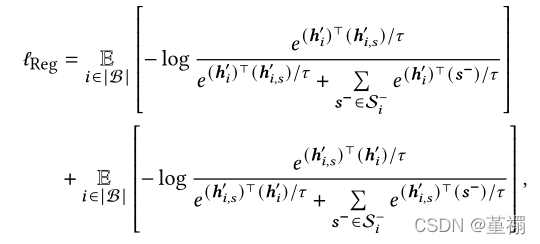


**Negative Sampling**

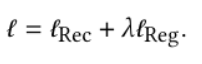
Để xây dựng hiệu quả các mẫu âm cho một cặp mẫu tăng cường, tất cả các mẫu tăng cường khác trong cùng một đợt huấn luyện đều được coi là mẫu âm tính. Giả sử rằng lô đào tạo là B và kích thước lô là |B|, sau khi tăng cường, sẽ có 2|B| các hidden vectơ, trong đó chỉ số dưới và chỉ số trên được quá tải để biểu thị chỉ mục của các mẫu trong lô và phần mở rộng cho rõ ràng. Do đó, đối với mỗi cặp dương trong lô, có 2(|B| − 1) cặp âm dưới dạng tập âm S −.

**Regularization Objective**

Tương tự như công thức NCE trước đó, sau khi có các mẫu dương và âm ở đây, một hàm mất mát (hàm thứ hai) cũng có thể được xây dựng như sau:

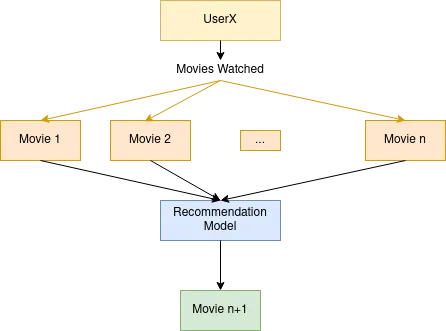


Tương tự như công thức NCE trước đó, sau khi có các mẫu dương và âm ở đây, một hàm mất mát tương tự (hàm thứ hai cũng) có thể được xây dựng, như sau:



## 2.2. Mô hình BERT4Rec

### 2.2.1. Giới thiệu chung



Việc mô hình hóa user linh hoạt từ lịch sử hành vi là một thách thức và quan trọng đối với các hệ thống recommend. Trước đó các phương pháp sử dụng Sequential Neural Network để encode lịch sử tương tác của người dùng từ trái sang phải thánh các hidden representation để đưa ra đề xuất. Nó tồn tại một số hạn chế bao gồm:

* Kiến trúc unidirectional hạn chế sức mạnh của hidden representation trong chuỗi hành vi của người dùng.
* Họ thường giả định một trình tự có thứ tự cứng nhắc, không phải lúc nào cũng thực tế.

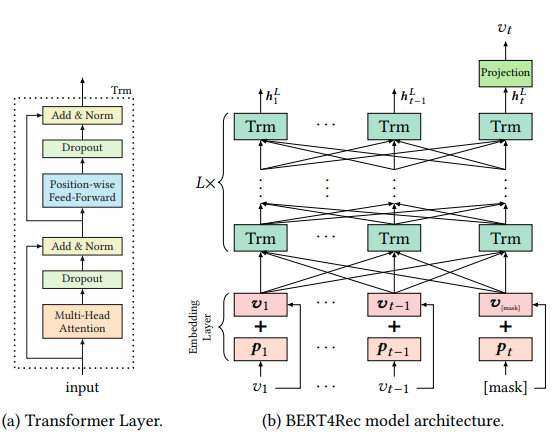
Để giải quyết những hạn chế này chúng tôi đã đề xuất một mô hình đề xuất tuần tự có tên là **BERT4Rec** sử dụng khả năng deep bidirectional self-attention để mô hình hóa các chuỗi hành vi của người dùng. Để tránh rò rỉ thông tin và đào tạo bidirectional model một cách hiệu quả, chúng tôi áp dụng mục tiêu Cloze cho đề xuất tuần tự, dự đoán các mục được che dấu ngẫu nhiên trong chuỗi bằng cách cùng điều chỉnh ngữ cảnh bên trái và bên phải của chúng. Bằng cách này, chúng tôi tìm hiểu mô hình biểu diễn hai chiều để đưa ra đề xuất bằng cách cho phép từng mục trong hành vi lịch sử của người dùng kết hợp thông tin từ cả bên trái và bên phải. Các thử nghiệm mở rộng trên bốn bộ dữ liệu chuẩn cho thấy mô hình của chúng tôi luôn vượt trội so với các mô hình tuần tự SOTA khác.

Việc mô tả chính xác sở thích của người dùng là trọng tâm của một hệ thống khuyến nghị hiệu quả. Trong nhiều ứng dụng trong thế giới thực, sở thích hiện tại của người dùng về bản chất là linh hoạt và phát triển, bị ảnh hưởng bởi các hành vi lịch sử của họ. Ví dụ: một người có thể mua phụ kiện (ví dụ: bộ điều khiển Joy-Con) ngay sau khi mua Nintendo Switch, mặc dù họ sẽ không mua phụ kiện bảng điều khiển trong các trường hợp bình thường

Để mô hình hóa các động lực tuần tự như vậy trong hành vi của người dùng, nhiều phương pháp đã được đề xuất để đưa ra các đề xuất tuần tự dựa trên các tương tác lịch sử của người dùng. Chúng nhằm mục đích dự đoán (các) mục liên tiếp mà người dùng có khả năng tương tác với các tương tác trước đây của cô ấy/anh ấy. Gần đây, một loạt các công việc sử dụng các mạng thần kinh tuần tự, ví dụ: Mạng thần kinh tái phát (RNN), để đề xuất tuần tự và thu được kết quả đầy hứa hẹn. Mô hình cơ bản của công việc trước đây là mã hóa các tương tác lịch sử của người dùng thành một vectơ (nghĩa là biểu thị sở thích của người dùng) bằng cách sử dụng mô hình tuần tự từ trái sang phải và đưa ra đề xuất dựa trên biểu diễn ẩn này.

Để mô hình hóa các động lực tuần tự như vậy trong hành vi của người dùng, nhiều phương pháp đã được đề xuất để đưa ra các đề xuất tuần tự dựa trên các tương tác lịch sử của người dùng. Chúng nhằm mục đích dự đoán (các) mục liên tiếp mà người dùng có khả năng tương tác với các tương tác trước đây của người dùng. Gần đây, một loạt các công việc sử dụng các mạng thần kinh tuần tự, ví dụ: Recurrent Neural Network (RNN), để đề xuất tuần tự và thu được kết quả đầy hứa hẹn. Mô hình cơ bản của công việc trước đây là encode các tương tác lịch sử của người dùng thành một vectơ (nghĩa là biểu thị sở thích của người dùng) bằng cách sử dụng mô hình left to right sequential model và đưa ra đề xuất dựa trên hidden representation này.

Hạn chế được mình họa như hình 1a, 1c là các mô hình unidirectional models với các mục trong chuỗi lịch sử, trong đó mỗi mục chỉ có thể mã hóa thông tin từ các mục trước đó Một hạn chế khác là các mô hình unidirectional models trước đây ban đầu được giới thiệu cho dữ liệu tuần tự với thứ tự tự nhiên, ví dụ: dữ liệu chuỗi văn bản và thời gian. Chúng thường giả định một trình tự được sắp xếp cứng nhắc trong dữ liệu, điều này không phải lúc nào cũng đúng đối với hành vi của người dùng trong các ứng dụng trong thế giới thực. Trên thực tế, các lựa chọn mục trong tương tác lịch sử của người dùng có thể không tuân theo giả định thứ tự cứng nhắc do nhiều yếu tố bên ngoài không thể quan sát được. Trong tình huống như vậy, điều quan trọng là phải kết hợp bối cảnh từ cả hai hướng trong mô hình trình tự hành vi của người dùng.



Hình 2.:

### 2.2.2. Những vấn đề liên quan

#### 2.2.2.1. Hệ thống recommendation tổng quan

Các nghiên cứu ban đầu về hệ thống đề xuất thường sử dụng Lọc cộng tác (CF) để mô hình hóa sở thích của người dùng dựa trên lịch sử tương tác của họ. Trong số các phương pháp CF khác nhau, Matrix Factorization (MF) là phương pháp phổ biến nhất, chiếu người dùng và vật phẩm vào một không gian vectơ dùng chung và ước tính sở thích của người dùng đối với một vật phẩm bằng tích bên trong giữa các vectơ của họ. Một dòng công việc khác là các phương pháp lân cận dựa trên mục. Họ ước tính mức độ ưa thích của người dùng đối với một mặt hàng thông qua đo lường mức độ tương đồng của mặt hàng đó với các mặt hàng trong lịch sử tương tác của họ bằng cách sử dụng ma trận tương tự giữa mặt hàng với mặt hàng được tính toán trước.

Gần đây, deep learning đã cách mạng hóa đáng kể các hệ thống khuyến nghị. Công việc tiên phong ban đầu là Restricted Boltzmann Machines (RBM) để lọc cộng tác, được đề xuất bởi Salakhutdinov trong Netflix Prize. Một dòng phương pháp dựa trên học sâu tìm cách cải thiện hiệu suất đề xuất bằng cách tích hợp các biểu diễn mục phân tán được học từ thông tin phụ trợ, ví dụ: văn bản, hình ảnh, các đặc trưng âm thanh vào các mô hình CF. Một số khác tìm cách thay đổi hệ số matrix factorization ví dụ như: Neural Collaborative Filtering (NCF) dự đoán sở thích qua Multi-Layer Perceptions (MLP) thay vì inner product, trong khi AutoRec and CDAE sử dụng Auto encoder framework.

#### 2.2.2.2. Hệ thống recommendation tuần tự

Thật không may, không có phương pháp nào ở trên dành cho đề xuất tuần tự vì tất cả chúng đều bỏ qua thứ tự trong hành vi của người dùng. Các công trình ban đầu về đề xuất tuần tự thường nắm bắt các mẫu tuần tự từ các tương tác lịch sử của người dùng bằng cách sử dụng chuỗi Markov (MC).

Bên cạnh các MC bậc nhất, các MC bậc cao cũng được sử dụng để xem xét các mục trước đó. Gần đây, RNN và các biến thể của nó, Gated Recurrent Unit (GRU) và Long Short-Term Memory (LSTM), ngày càng trở nên phổ biến để mô hình hóa các chuỗi hành vi của người dùng. Ý tưởng cơ bản của các phương pháp này là encode các bản ghi trước đó của người dùng thành một véc-tơ (tức là biểu diễn sở thích của người dùng được sử dụng để đưa ra dự đoán) với các kiến trúc lặp lại và hàm loss khác nhau, bao gồm GRU with ranking loss (GRU4Rec), Dynamic REcurrent bAsket Model (DREAM), user-based GRU,attention-based GRU (NARM) và GRU4Rec cải tiến với hàm loss mới (tức là, BPR-max và TOP1-max) và một chiến lược cải tiến.

Ngoài các RNN, các mô hình học sâu khác nhau cũng được giới thiệu để đề xuất tuần tự. Ví dụ, Convolutional Sequence Model (Caser) để học các mẫu trình tự sử dụng cả bộ lọc tích chập ngang và dọc, sử dụng Memory Network để cải thiện đề xuất tuần tự. STAMP nắm bắt cả sở thích chung và sở thích hiện tại của người dùng bằng cách sử dụng mạng MLP với sự chú ý.

#### 2.2.2.3. Cơ chế Attention

Cơ chế Attention đã cho thấy tiềm năng đầy hứa hẹn trong việc mô hình hóa dữ liệu tuần tự, ví dụ:machine translation và text classification. Gần đây, một số nghiên cứu cố gắng sử dụng cơ chế Attention để cải thiện hiệu suất đề xuất và khả năng diễn giải. Ví dụ: kết hợp cơ chế Attention vào GRU để nắm bắt cả hành vi tuần tự của người dùng và mục đích chính trong session-based recommendation.

Các nghiên cứu được đề cập ở trên về cơ bản coi cơ chế chú ý như một thành phần bổ sung cho các mô hình ban đầu. Ngược lại, Transformer và BERT chỉ được xây dựng dựa trên khả năng tự chú ý của nhiều đầu và đạt được kết quả tiên tiến nhất về mô hình chuỗi văn bản. Gần đây, ngày càng có nhiều người nhiệt tình áp dụng các attention-based neural networks để mô hình hóa dữ liệu tuần tự về tính hiệu quả và hiệu quả của chúng. Đối với đề xuất tuần tự, Kang và McAuley giới thiệu bộ giải mã Transformer hai lớp (tức là mô hình ngôn ngữ Transformer) có tên là SASRec để nắm bắt các hành vi tuần tự của người dùng và đạt được kết quả hiện đại trên một số bộ dữ liệu công khai. SASRec có liên quan mật thiết đến công việc của chúng tôi. Tuy nhiên, nó vẫn là một mô hình đơn hướng sử dụng attention mask thông thường. Mặc dù chúng tôi sử dụng mô hình hai chiều để mã hóa chuỗi hành vi của người dùng với sự trợ giúp của tác vụ Cloze.

### 2.2.3. Kiến trúc mô hình BERT4Rec

#### 2.2.3.1 Vấn đề Statement

Trong bài toán đề xuất tuần tự, đặt **U**={u1,u2, . . . ,u|**U** | }, chỉ tập hợp user, **V={**v1,v2, . . . ,v|**V** | } là tập hợp các item, và list Su=[v1(u) , . . . ,vt(u) , . . . , ] là các tương tác tuần tự trong thứ tự thời gian cho các user u ∈**U,** trong đó vt(u) ∈**V** là item mà user u đã tương trong khoảng thời gian t và nu là độ dài của tương tác tuần tự cho user u. Với lịch sự tương tác Su, đề xuất tuần tự tập trung vào item và user sẽ tương tác tại thời điểm nu + 1. Nó có thể biểu diễn dưới dạng xác suất item cho user tại thời điểm nu + 1 như sau;



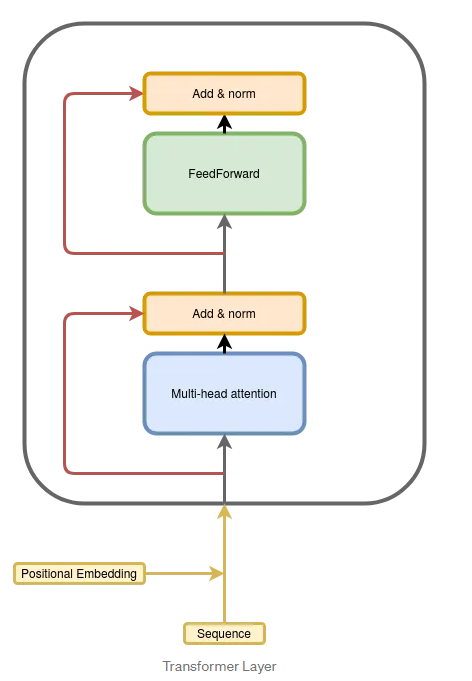
#### 2.2.3.2. Kiến trúc mô hình tổng quan

Ở đây, chúng tôi giới thiệu một mô hình đề xuất tuần tự mới được gọi là BERT4Rec, mô hình này sử dụng Bidirectional Encoder Representations từ Transformer cho một task mới, Đề xuất tuần tự. Nó được xây dựng dựa trên lớp self-attention phổ biến “Transformer layer”.

Như được minh họa trong Hình 1b, BERT4Rec được xếp chồng lên nhau bởi L bidirectional Transformer layers. Tại mỗi lớp, nó sửa đổi lặp đi lặp lại biểu diễn của mọi vị trí bằng cách trao đổi thông tin trên tất cả các vị trí ở lớp trước song song với lớp Transformer. Thay vì học cách chuyển từng bước thông tin có liên quan về phía trước như các phương pháp dựa trên RNN đã làm trong Hình 1d, cơ chế self-attention cung cấp cho BERT4Rec khả năng nắm bắt trực tiếp các phần phụ thuộc ở bất kỳ khoảng cách nào. Cơ chế này dẫn đến trường tiếp nhận toàn cục, trong khi các phương pháp dựa trên CNN như Caser thường có trường tiếp nhận hạn chế. Ngoài ra, trái ngược với các phương pháp dựa trên RNN, tự chú ý rất dễ thực hiện song song.

So sánh Hình 1b, 1c và 1d, sự khác biệt đáng chú ý nhất là các phương thức dựa trên SASRec và RNN đều là kiến trúc đơn hướng từ trái sang phải, trong khi BERT4Rec của chúng tôi sử dụng tính năng s bidirectional self-attention để lập mô hình trình tự hành vi của người dùng. Bằng cách này, mô hình được đề xuất của chúng tôi có thể thu được các biểu diễn mạnh mẽ hơn về chuỗi hành vi của người dùng để cải thiện hiệu suất đề xuất.

#### 2.2.3.3. Transformer Layer



Như trong hình 1b, với input độ dài tuần tự là t, chúng ta tính toán hidden representation tại mỗi layer l cho vị trí i bằng cách áp dụng lớp transformer. Tại đây sẽ xếp chồng với nhau thành ma trận vì chúng tôi tính toán hàm attention đồng thời trên tất cả các vị trí trong thực tế. Như trong Hình 1a, lớp Transformer *Trm* chứa hai lớp con, lớp con Multi-Head Self-Attention và Position-wise Feed-Forward Network.

**Multi-Head Self-Attention:** Các cơ chế attention đã trở thành một phần không thể thiếu của mô hình hóa trình tự trong nhiều nhiệm vụ khác nhau, cho phép nắm bắt sự phụ thuộc giữa các cặp representation mà không quan tâm đến khoảng cách của chúng trong trình tự. Nghiên cứu trước đây đã chỉ ra rằng sẽ có lợi khi cùng tham gia vào thông tin từ các không gian con biểu diễn khác nhau ở các vị trí khác nhau. Do đó, ở đây chúng tôi áp dụng multi-head self-attention thay vì thực hiện single attention function. Cụ thể, trước tiên,multi-head attention chiếu tuyến tính vào các không gian con *h*, với các phép chiếu tuyến tính khác nhau, có thể học được, sau đó áp dụng hàm attention *h* song song để tạo ra các representation đầu ra được nối và chiếu lại một lần nữa:

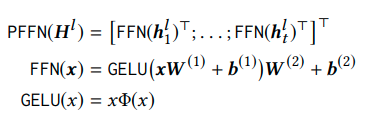
(1)

Trong dó,projection matrix cho mỗi head , là tham số có thể học. Ở đây, chúng tôi bỏ qua chỉ số l để đơn giản . Trong thực tế, tham số chiếu này không được chia sẻ qua lớp. Ở đó, hàm Attention là Scaled Dot-Product Attention:

(2)

Trong đó query Q, key K, và value V là hình chiếu từ ma trận với ma trận chiếu được học khác nhau như trong công thức (1). Temperature được giới thiệu để tạo ra phân bố attention để tránh vanishing gradient.

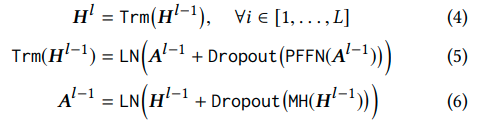
**Position-wise Feed-Forward Network:** Như mô tả ở trên, self-attention được dựa trên phép chiếu tuyến tính. Để cung cấp cho mô hình tính phi tuyến tính và tương tác giữa các thứ nguyên khác nhau, chúng tôi áp dụng Position-wise Feed-Forward Network theo vị trí cho các đầu ra của lớp phụ self-attention, riêng biệt và giống hệt nhau ở mỗi vị trí. Nó bao gồm hai phép biến đổi affine với kích hoạt Gaussian Error Linear Unit (GELU) ở giữa:

 (3)

Trong đó Ф(x) là hàm phân phối tích lũy của phân phối Gauss, , , , . là tham số có thể học và được chia sẻ giữa các vị trí. Chúng tôi bỏ qua lớp l để thuận tiện hơn. Trên thực thế những tham số này là khác biệt từ lớp này đến lớp khác. Ở đây, theo OPEN AI GPT và BERT, chúng tôi sử dụng smoother GELU activation hơn là standard RELU activation.

**Stacking Transformer Layer:** Như đã trình bày chi tiết ở trên, chúng ta có thể dễ dàng nắm bắt các tương tác giữa item và item trên toàn bộ chuỗi hành vi của người dùng bằng cơ chế tự chú ý. Tuy nhiên, việc tìm hiểu các mẫu chuyển đổi vật phẩm phức tạp hơn bằng cách xếp chồng các lớp tự chú ý thường có lợi. Tuy nhiên, mạng trở nên khó đào tạo hơn khi đi sâu hơn. Do đó, chúng tôi sử dụng y a residual connection xung quanh mỗi lớp trong số hai lớp con như trong Hình 1a, tiếp theo là layer normalization. Hơn nữa, chúng tôi cũng áp dụng dropout cho đầu ra của mỗi lớp con, trước khi nó được chuẩn hóa. Nghĩa là, đầu ra của mỗi lớp con là LN(x + Dropout(lớp con(x))), trong đó lớp con(·) là chức năng do chính lớp con thực hiện, LN là hàm chuẩn hóa lớp được định nghĩa trong [1 ]. Chúng tôi sử dụng LN để chuẩn hóa các đầu vào trên tất cả các đơn vị ẩn trong cùng một lớp để ổn định và tăng huấn luyện.

Tuy nhiên, BERT4Rec tinh chỉnh các hidden representation của mỗi lớp như sau:



#### 2.2.3.4. Embedding Layer

Như đã giải thích ở trên, không có bất kỳ mô-đun lặp hoặc tích chập nào, lớp transformer Trm không nhận thức được thứ tự của đầu vào sự liên tiếp. Để sử dụng thông tin tuần tự của đầu vào, chúng tôi đưa các Positional Embed vào các embedding mục đầu vào ở dưới cùng của stack lớp Transformer. Đối với một mục nhất định vi , đại diện đầu vào của nó được mô tả liên quan giữa item và positional embedding:

Trong đó là embedding d chiều cho item là positional embedding d chiều cho vị trí index thứ i. Ở đây, chúng tôi sử dụng the learnable positional embedding thay vì sinusoid embedding cố định để cho hiệu năng cao hơn. Positional embedding cho phép mô hình của chúng tôi xác định phần đầu vào mà nó đang xử lý. Tuy nhiên, nó cũng áp đặt một hạn chế về độ dài câu tối đa N mà mô hình của chúng tôi có thể xử lý. DO đó chúng tôi cần cắt bớt chuỗi đầu vào [v1, …, vt]. tới N item cuối cùng nếu t > **N**.

#### 2.2.3.5. Output Layer

Sau L lớp trao đổi thông tin theo thứ bậc trên tất cả các vị trí trong lớp trước đó, chúng tôi nhận được HL đầu ra cuối cùng cho tất cả các mục của chuỗi đầu vào. Giả sử rằng chúng ta mask vật phẩm vt tại thời điểm bước t, sau đó chúng ta dự đoán các vật phẩm mask vt dựa trên như trong Hình 1b. Cụ thể, chúng tôi áp dụng mạng chuyển tiếp nguồn cấp dữ liệu hai lớp với kích hoạt GELU ở giữa để tạo phân phối đầu ra trên mục tiêu:

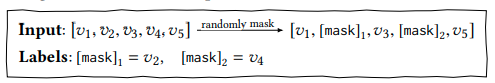


Trong đó **WP** là ma trận chiếu có thể học, **bP và bO** là thành phần bias, là ma trận embedding cho tập item 𝓥. Chúng tôi chia sẻ ma trận embedding iem trong lớp đầu vào và đầu ra để giảm hiện tượng overfitting và giảm dung lượng model.

#### 2.2.3.6 Quá trình huấn luyện

Các mô hình unidirectional sequential recommendation thông thường train bằng cách dự đoán item tiếp theo cho mỗi vị trí trong mỗi chuỗi đầu vào như hình 1c và 1d. Đặc biệt, mục tiêu của chuỗi đầu vào [v1, …, vt] là dịch thành một chuỗi khác [v2, …, vt+1]. Tuy nhiên như hình 1b cùng điều chỉnh cả ngữ cảnh trái và phải trong bidirectional model sẽ khiến biểu diễn đầu ra cuối cùng của mỗi mục chứa thông tin của mục đích. Điều này làm cho việc dự đoán tương lai trở nên tầm thường và mạng sẽ không học được điều gì hữu ích. Một giải pháp đơn giản cho vấn đề này là tạo t - 1 chuỗi con với các mục tiếp theo như ([v1], v2) và ([v1,v2], v3)) từ độ dài ban đầu của chuỗi hành vi t và sau đó mã hóa từng chuỗi con lịch sử bằng bidirectional model để dự đoán. Tuy nhiên, cách tiếp cận này rất tốn thời gian và tài nguyên vì chúng ta cần tạo một mẫu mới cho từng vị trí trong chuỗi và dự đoán chúng một cách riêng biệt.

Để đào tạo mô hình được đề xuất một cách hiệu quả, chúng tôi áp dụng một mục tiêu mới: *Cloze task* (còn được gọi là “Masked Language Model” ) cho đề xuất tuần tự. Đây là một bài kiểm tra bao gồm một phần ngôn ngữ đã bị loại bỏ một số từ, trong đó người tham gia được yêu cầu điền các từ còn thiếu. Trong trường hợp của chúng tôi, đối với mỗi bước đào tạo, chúng tôi ẩn ngẫu nhiên *ρ* tỷ lệ của tất cả các mục trong chuỗi đầu vào (nghĩa là thay thế bằng special token “[mask]”), sau đó dự đoán id ban đầu của các mục bị che chỉ dựa trên bên trái của nó và đúng ngữ cảnh. Ví dụ:



Các vectơ ẩn cuối cùng tương ứng với “[mask]” được đưa vào softmax đầu ra trên tập hợp item, như trong đề xuất tuần tự thông thường. Cuối cùng, chúng tôi xác định loss cho mỗi đầu vào masked t S ′ u là negative log-likelihood của các marked target:



Trong đó là phiên bản masked của lịch sử hành vi , là mục bị ẩn ngẫu nhiên trong nó, là item thực tế cho item bị ẩn và xác suất P(.) được định nghĩa trong công thức 7.

Một lợi thế bổ sung cho Cloze task là nó có thể tạo ra nhiều mẫu hơn để huấn luyện mô hình. Giả sử một chuỗi có độ dài n, các dự đoán tuần tự thông thường trong Hình 1c và 1d tạo ra n mẫu duy nhất để đào tạo, trong khi BERT4Rec có thể lấy n mẫu (nếu chúng ta che k mục ngẫu nhiên) trong nhiều epochs. Nó cho phép chúng tôi đào tạo một mô hình bidirectional representation model mạnh mẽ hơn.

## 2.3. Mô hình LSTM

# III.  Đề xuất và xây dựng nhóm mô hình Machine Learning

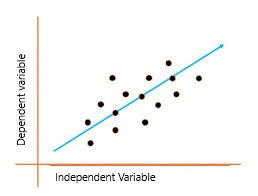
## 3.1. Mô hình Linear Regression

### 3.1.1. Tổng quan về mô hình Linear Regression

Linear Regression (hồi quy tuyến tính) là một thuật toán học có giám sát (*Supervised learning),* đây là một trong những thuật toán cơ bản và dễ dàng triển khai nhất của của học maý (Machine Learning).

Hồi quy tuyến tính là một loại phân tích thống kê được sử dụng để dự đoán mối quan hệ giữa hai biến. Nó giả định mối quan hệ tuyến tính giữa biến độc lập và biến phụ thuộc, mục đích là tìm ra phương trình đường thẳng phù hợp nhất để mô tả mối quan hệ đó. Đường thẳng biểu diễn được xác định bằng cách giảm thiểu tổng bình phương chênh lệch giữa giá trị dự đoán và giá trị thực. Hồi quy tuyến tính thường được sử dụng trong nhiều lĩnh vực, bao gồm kinh tế, tài chính và khoa học xã hội, để phân tích và dự đoán xu hướng của dữ liệu. Nó cũng có thể được mở rộng cho hồi quy tuyến tính bội, trong đó có nhiều biến độc lập và hồi quy logistic, được sử dụng cho các bài toán phân loại nhị phân.

Một mô hình hồi quy tuyến tính đơn giản bao gồm một biến độc lập và một biến phụ thuộc. Mô hình ước tính độ dốc và giao điểm của đường phù hợp nhất, qua đó biểu thị mối quan hệ giữa các biến. Độ dốc biểu thị sự thay đổi của biến phụ thuộc đối với mỗi thay đổi đơn vị trong biến độc lập, trong khi hệ số chặn biểu thị giá trị dự đoán của biến phụ thuộc khi biến độc lập bằng 0. Hồi quy tuyến tính là một phương pháp hồi quy thống kê đơn giản nhất được sử dụng để phân tích dự đoán trong học máy. Hồi quy tuyến tính cho thấy mối quan hệ tuyến tính giữa biến độc lập (bộ dự báo) tức là trục X và biến phụ thuộc (đầu ra) tức là trục Y, được gọi là hồi quy tuyến tính. Nếu có một biến đầu vào duy nhất X (biến độc lập), hồi quy tuyến tính như vậy được gọi là hồi quy tuyến tính đơn giản.



Biểu đồ trên trình bày mối quan hệ tuyến tính giữa biến đầu ra (y) và biến dự đoán (X). Đường màu xanh được gọi là đường thẳng tốt nhất. Dựa trên các điểm dữ liệu đã cho, tôi cố gắng vẽ một đường phù hợp nhất với các điểm đó.

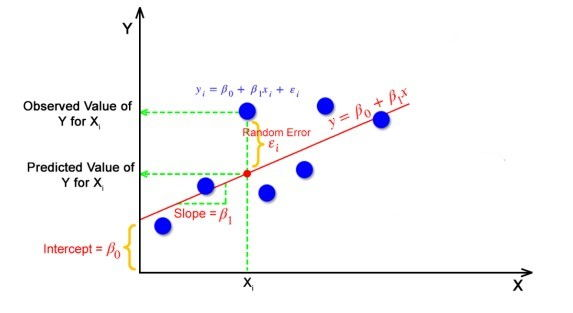
Để tìm ra đường biểu diễn quan hệ phù hợp nhất, hồi quy tuyến tính áp dụng biểu mẫu chặn độ dốc truyền thống được đưa ra dưới đây:

**Yi = β0 + β1Xi**

trong đó:

* Yi = Biến phụ thuộc
* β0 = hằng số/Giao điểm
* β1 = Hệ số góc/Giao điểm
* Xi = Biến độc lập

Thuật toán này giải thích mối quan hệ tuyến tính giữa biến phụ thuộc (đầu ra) y và biến độc lập (bộ dự đoán) X bằng cách sử dụng một phương trình đường thẳng Y= B0 + B1X.



Mục tiêu của thuật toán hồi quy tuyến tính là lấy các giá trị tốt nhất cho B0 và B1 để tìm đường biểu diễn quan hệ phù hợp nhất. Đường phù hợp nhất là đường có ít lỗi nhất, nghĩa là lỗi giữa giá trị dự đoán và giá trị thực tế phải nhỏ nhất.

Trong hồi quy, chênh lệch giữa giá trị của biến phụ thuộc (yi) và giá trị dự đoán (predicted) được gọi là phần dư.

**εi =**  **ypredicted** – **yi**

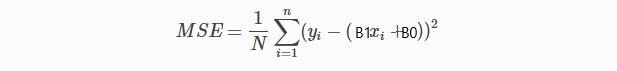
trong đó: ypredicted = B0 + B1 Xi

Nói một cách đơn giản, đường biểu diễn tốt nhất là đường thể hiện tốt nhất với tập dữ liệu đã cho. Về mặt toán học, đường phù hợp nhất thu được bằng cách minimize tổng phần dư được trình bày phía trên.

### 3.1.2. Các vấn đề liên quan

#### 3.1.2.1 Loss function (Hàm mất mát)

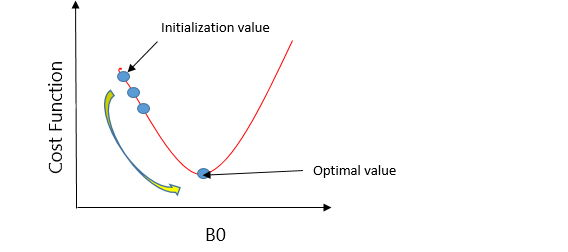
Hàm chi phí giúp tìm ra các giá trị tối ưu cho B0 và B1, qua đó tìm ra đường phù hợp nhất cho các điểm dữ liệu. Trong Hồi quy tuyến tính, hàm mất mát Mean Squared Error (MSE) thường được sử dụng, hàm này là giá trị trung bình của sai số bình phương giữa ypredicted và yi.



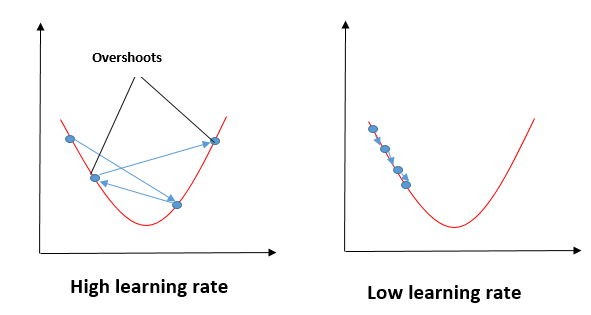
Sử dụng hàm MSE, ta sẽ cập nhật các giá trị của B0 và B1 sao cho giá trị MSE ổn định ở mức cực tiểu. Các tham số này có thể được xác định bằng phương pháp giảm dần độ dốc (gradient descent) sao cho giá trị của hàm chi phí là nhỏ nhất.

#### 3.1.2.2 Gradient Descent trong Linear Regression

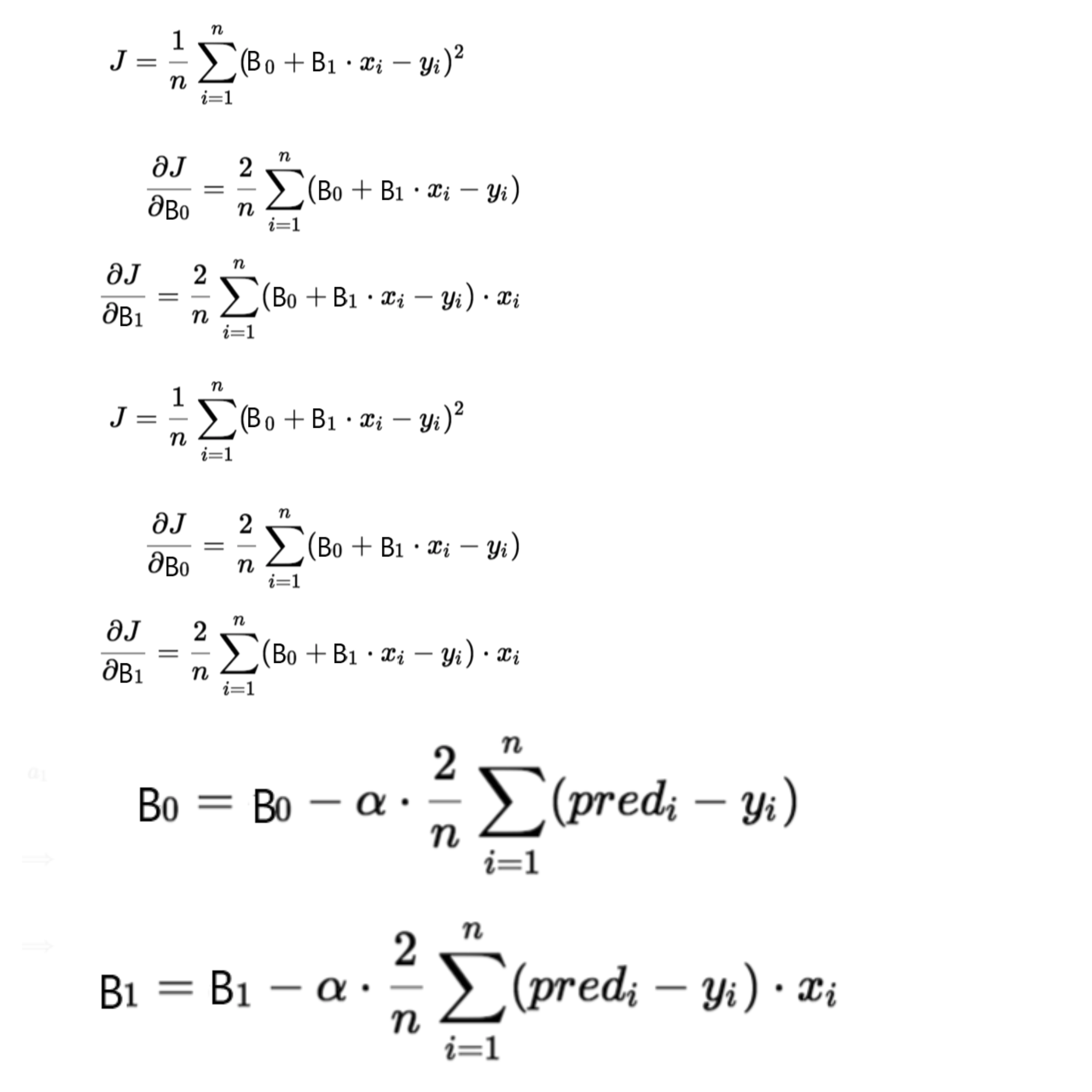
Gradient Descent là một trong những thuật toán tối ưu hóa hàm mất mát (hàm mục tiêu) để đạt được giải pháp tối ưu cho bài toán. Để tìm giải pháp tối ưu, chúng ta cần giảm hàm chi phí (MSE) cho tất cả các điểm dữ liệu, điều này được thực hiện bằng cách cập nhật lặp đi lặp lại các giá trị của B0 và B1. Mô hình hồi quy tối ưu hóa thuật toán Gradient Descent để cập nhật các hệ số của đường thẳng bằng cách giảm hàm mất mát thông qua việc chọn ngẫu nhiên các giá trị hệ số và sau đó cập nhật lặp lại các giá trị để đạt được hàm mất mát tối thiểu.



Xét ví dụ như sau. Hãy tưởng tượng một hố hình chữ U và ta đang đứng ở điểm cao nhất trong hố, và mục đích của ta là chạm tới đáy hố. Giả sử có một kho báu ở dưới đáy hố, và ta chỉ có thể đi một số bước riêng biệt để xuống tới đáy. Nếu ta chọn thực hiện từng bước một, cuối cùng bạn sẽ đến được đáy hố, nhưng điều này sẽ mất nhiều thời gian hơn. Nếu ta quyết định thực hiện các bước lớn hơn mỗi lần, bạn có thể chạm đáy sớm hơn, nhưng có khả năng ta có thể vượt qua đáy của hố và thậm chí không ở gần đáy. Trong thuật toán giảm Gradient Descent, số bước ta đang thực hiện có thể được coi là tốc độ học (learning rate) và điều này quyết định tốc độ hội tụ của thuật toán đến cực tiểu.



Để cập nhật B0 và B1, ta lấy đạo hàm từ hàm chi phí. Để tìm các B0 và B1 này, chúng ta lấy đạo hàm riêng của B0 và B1.



Chúng ta cần giảm thiểu hàm mất mát J. Một trong những cách để đạt được điều này là áp dụng thuật toán *Batch Gradient Descent*. Trong *Batch Gradient Descent*, các giá trị được cập nhật trong mỗi lần lặp lại. (Hai phương trình cuối cho thấy việc cập nhật các giá trị) Các đạo hàm riêng là các gradient và chúng được sử dụng để cập nhật các giá trị của B0 và B1. Alpha là learning rate.

#### 3.1.2.3 Phương thức đánh giá mô hình Linear Regression

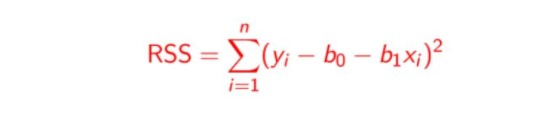
Kết quả của bất kỳ mô hình hồi quy tuyến tính nào có thể được đánh giá bằng nhiều độ đo khác nhau. Các độ đo này thường cung cấp thước đo về mức độ hiệu quả của các đầu ra quan sát được do mô hình tạo ra. Các độ đo được sử dụng nhiều nhất là Coefficient of Determination or R-Squared (R2), Root Mean Squared Error (RSME) và Residual Standard Error (RSE).

**Coefficient of Determination or R-Squared (R2)**

R-Squared là một con số giải thích lượng biến thể được ghi lại bởi mô hình đã phát triển. Nó luôn nằm trong khoảng từ 0 & 1. Nhìn chung, giá trị của R-squared càng cao thì mô hình càng phù hợp với dữ liệu.

Về mặt toán học, nó có thể được biểu diễn dưới dạng:  **R2 = 1 – ( RSS/TSS )**

* **Residual sum of Squares (RSS)** được định nghĩa là tổng bình phương của phần dư cho mỗi điểm dữ liệu trong biểu đồ. Nó là thước đo sự khác biệt giữa đầu ra dự kiến và đầu ra thực tế quan sát được.

****

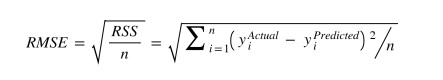
* **Total Sum of Squares (TSS)** được định nghĩa là tổng sai số của các điểm dữ liệu từ giá trị trung bình của biến dự đoán. Về mặt toán học TSS là:



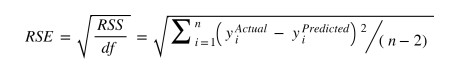
trong đó y mũ là giá trị trung bình của các điểm dữ liệu mẫu.

**Root Mean Squared Error**

Root Mean Squared Error là căn bậc hai của phương sai của phần dư. Nó chỉ định mức độ phù hợp tuyệt đối của mô hình với dữ liệu, tức là mức độ gần của các điểm dữ liệu được quan sát với các giá trị dự đoán. Về mặt toán học, nó có thể được biểu diễn dưới dạng:



Để làm cho ước tính này không chệch, người ta phải chia tổng bình phương phần dư cho bậc tự do chứ không phải tổng số điểm dữ liệu trong mô hình. Thuật ngữ này sau đó được gọi là **Residual Standard Error(RSE)**. Về mặt toán học, nó có thể được biểu diễn dưới dạng:



R-squared là thước đo tốt hơn RSME. Vì giá trị của Root Mean Squared Error phụ thuộc vào đơn vị của các biến (nghĩa là nó không phải là thước đo được chuẩn hóa), nó có thể thay đổi cùng với sự thay đổi trong đơn vị của các biến.

## 3.2. Mô hình Logistic Regression

### 3.2.1. Tổng quan về mô hình Logistic Regression

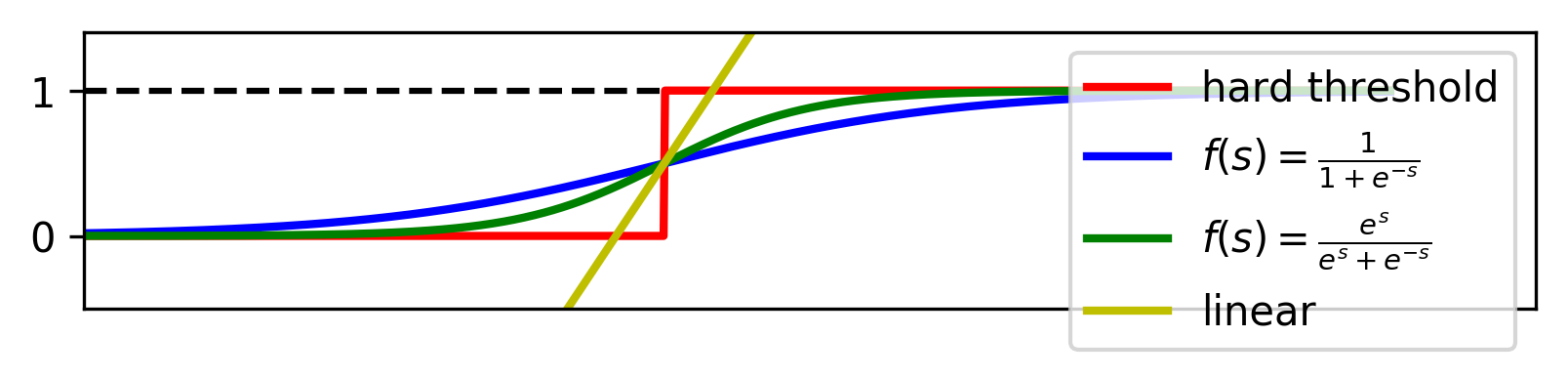
Hồi quy logistic (logistic regression) là một thuật toán học máy có giám sát, chủ yếu được sử dụng cho các nhiệm vụ phân loại trong đó mục tiêu là dự đoán xác suất một mẫu có thuộc về một lớp nhất định hay không. Nó là một loại thuật toán thống kê, phân tích mối quan hệ giữa một tập hợp các biến độc lập và các biến nhị phân phụ thuộc. Nó là một công cụ mạnh mẽ để ra quyết định. Ví dụ email có spam hay không.

Các thuật ngữ liên quan đến hồi quy logistic:

* Các biến độc lập (*Independent variables***)**: Các đặc điểm đầu vào hoặc các yếu tố dự đoán được áp dụng cho các dự đoán của biến phụ thuộc.
* Biến phụ thuộc (*Dependent variable*): Biến mục tiêu trong mô hình hồi quy logistic mà chúng tôi đang cố gắng dự đoán.
* Hàm logistic: Công thức được sử dụng để biểu thị mối quan hệ của các biến độc lập và biến phụ thuộc với nhau như thế nào. Hàm logistic biến đổi các biến đầu vào thành giá trị xác suất trong khoảng từ 0 đến 1, biểu thị khả năng biến phụ thuộc là 1 hoặc 0.
* Odds: Đó là tỷ lệ của một cái gì đó xảy ra với một cái gì đó không xảy ra. nó khác với xác suất vì xác suất là tỷ lệ của một điều gì đó xảy ra với mọi thứ có thể xảy ra.
* Log-odds: Tỷ lệ log, còn được gọi là hàm logit, là logarit tự nhiên của tỷ lệ cược. Trong hồi quy logistic, tỷ lệ chênh lệch log của biến phụ thuộc được mô hình hóa dưới dạng kết hợp tuyến tính của các biến độc lập và hệ số chặn.
* Hệ số: Các tham số ước tính của mô hình hồi quy logistic, cho biết mối quan hệ của các biến độc lập và biến phụ thuộc với nhau như thế nào.
* Intercept: Một thuật ngữ không đổi trong mô hình hồi quy logistic, đại diện cho tỷ lệ cược log khi tất cả các biến độc lập bằng 0.
* Maximum likelihood estimation**:** Phương pháp được sử dụng để ước tính các hệ số của mô hình hồi quy logistic, giúp tối đa hóa khả năng quan sát của dữ liệu mà mô hình đưa ra.

Đầu ra dự đoán của logistic regression thường được viết chung dưới dạng:

Trong đó θ được gọi là logistic function. Một số activation cho mô hình tuyến tính được cho trong hình dưới đây:



### 

### 3.2.2. Các vấn đề liên quan

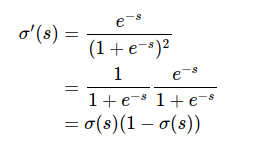
#### 3.2.2.1 Hàm Sigmoid

Trong Linear Regression, hàm activation được sử dụng nhiều nhất là hàm sigmoid vì đầu ra của nó được chặn trong khoảng (0, 1):



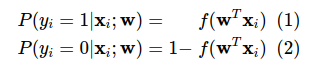


Đặc biệt hơn nữa:



#### 3.2.2.2 Hàm mất mát (Loss function)

Với mô hình như trên (các activation màu xanh lam và lục), ta có thể giả sử rằng xác suất để một điểm dữ liệu x rơi vào class 1 là và rơi vào class 0 là . Với mô hình được giả sử như vậy, với các điểm dữ liệu training (đã biết đầu ra y), ta có thể viết như sau:



trong đó được hiểu là xác suất xảy ra sự kiện đầu ra yi=1 khi biết tham số mô hình w và dữ liệu đầu vào xi. Mục đích của chúng ta là tìm các hệ số w sao cho càng gần với 1 càng tốt với các điểm dữ liệu thuộc class 1 và càng gần với 0 càng tốt với những điểm thuộc class 0.

Ký hiệu và viết gộp lại hai biểu thức bên trên ta có:



Biểu thức này là tương đương với hai biểu thức (1) và (2) ở trên vì khi yi=1, phần thứ hai của vế phải sẽ triệt tiêu, khi yi=0, phần thứ nhất sẽ bị triệt tiêu! Chúng ta muốn mô hình gần với dữ liệu đã cho nhất, tức xác suất này đạt giá trị cao nhất.

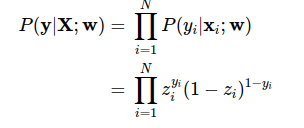
Xét toàn bộ training set với ∈ R*dxN* và , chúng ta cần tìm w để biểu thức sau đây đạt giá trị lớn nhất:



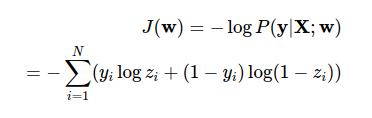
ở đây, ta cũng ký hiệu X,y như các [biến ngẫu nhiên](https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn_ng%E1%BA%ABu_nhi%C3%AAn) (random variables). Nói cách khác:



Giả sử thêm rằng các điểm dữ liệu được sinh ra một cách ngẫu nhiên độc lập với nhau (independent), ta có thể viết:



Trực tiếp tối ưu hàm số này theo w nhìn qua không đơn giản! Hơn nữa, khi N lớn, tích của N số nhỏ hơn 1 có thể dẫn tới sai số trong tính toán (numerical error) vì tích là một số quá nhỏ. Một phương pháp thường được sử dụng đó là lấy logarit tự nhiên (cơ số e) của *likelihood function* biến phép nhân thành phép cộng và để tránh việc số quá nhỏ. Sau đó lấy ngược dấu để được một hàm và coi nó là hàm mất mát. Lúc này bài toán tìm giá trị lớn nhất (maximum likelihood) trở thành bài toán tìm giá trị nhỏ nhất của hàm mất mát (hàm này còn được gọi là negative log likelihood):



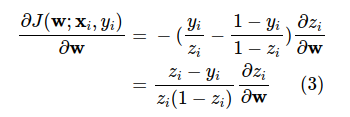
với chú ý rằng zi là một hàm số của w. Trong bài toán đang xét, một phân phối là dữ liệu được cho, với xác suất chỉ là 0 hoặc 1; phân phối còn lại được tính theo mô hình logistic regression. *Khoảng cách* giữa hai phân phối nhỏ đồng nghĩa với việc (*có vẻ hiển nhiên là*) hai phân phối đó rất gần nhau.

#### 3.2.2.3 Phương pháp tối ưu (Optimization)

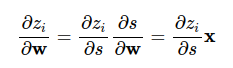
Chúng ta sử dụng phương pháp [Stochastic Gradient Descent](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#-stochastic-gradient-descent) (SGD), hàm mất mát với chỉ một điểm dữ liệu (xi, yi) là:



với đạo hàm:



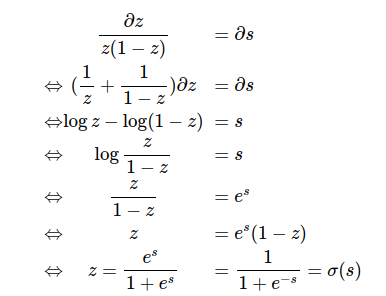
Để cho biểu thức này trở nên *gọn* và *đẹp* hơn, chúng ta sẽ tìm hàm sao cho mẫu số bị triệt tiêu. Nếu đặt , chúng ta sẽ có:



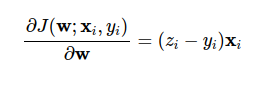
Một cách trực quan nhất, ta sẽ tìm hàm số sao cho:



để triệt tiêu mẫu số trong biểu thức (3). Chúng ta cùng khởi động một chút với phương trình vi phân đơn giản này. Phương trình (4) tương đương với:



Công thức cập nhật cho logistic sigmoid regression



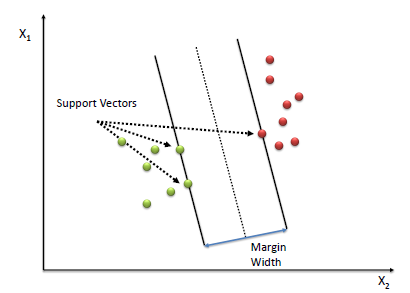
Và công thức cập nhật (theo thuật toán [SGD](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#-stochastic-gradient-descent)) cho logistic regression là:



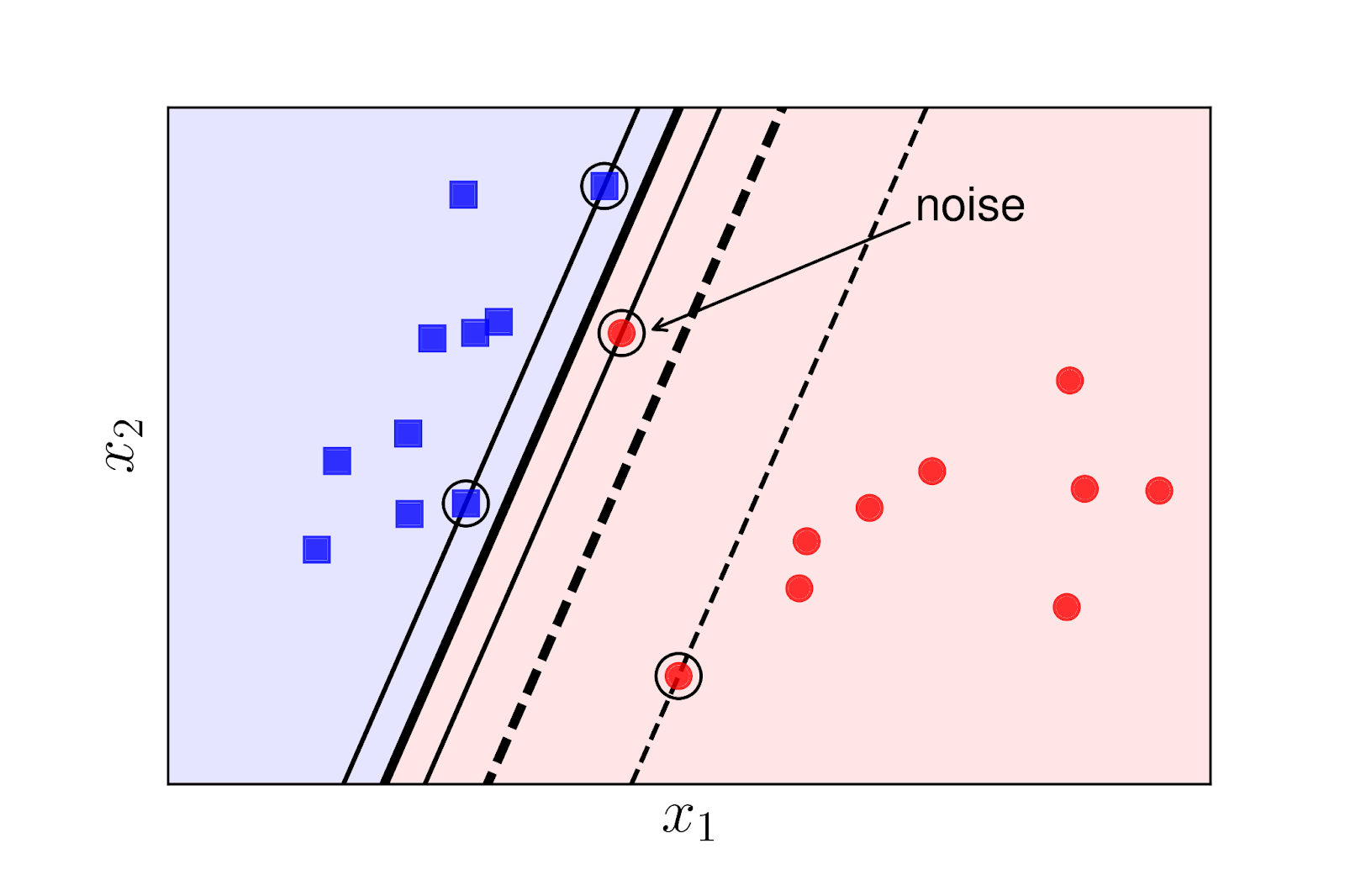
## 3.2. Mô hình SVM

### 3.2.1. Tổng quan về mô hình SVM

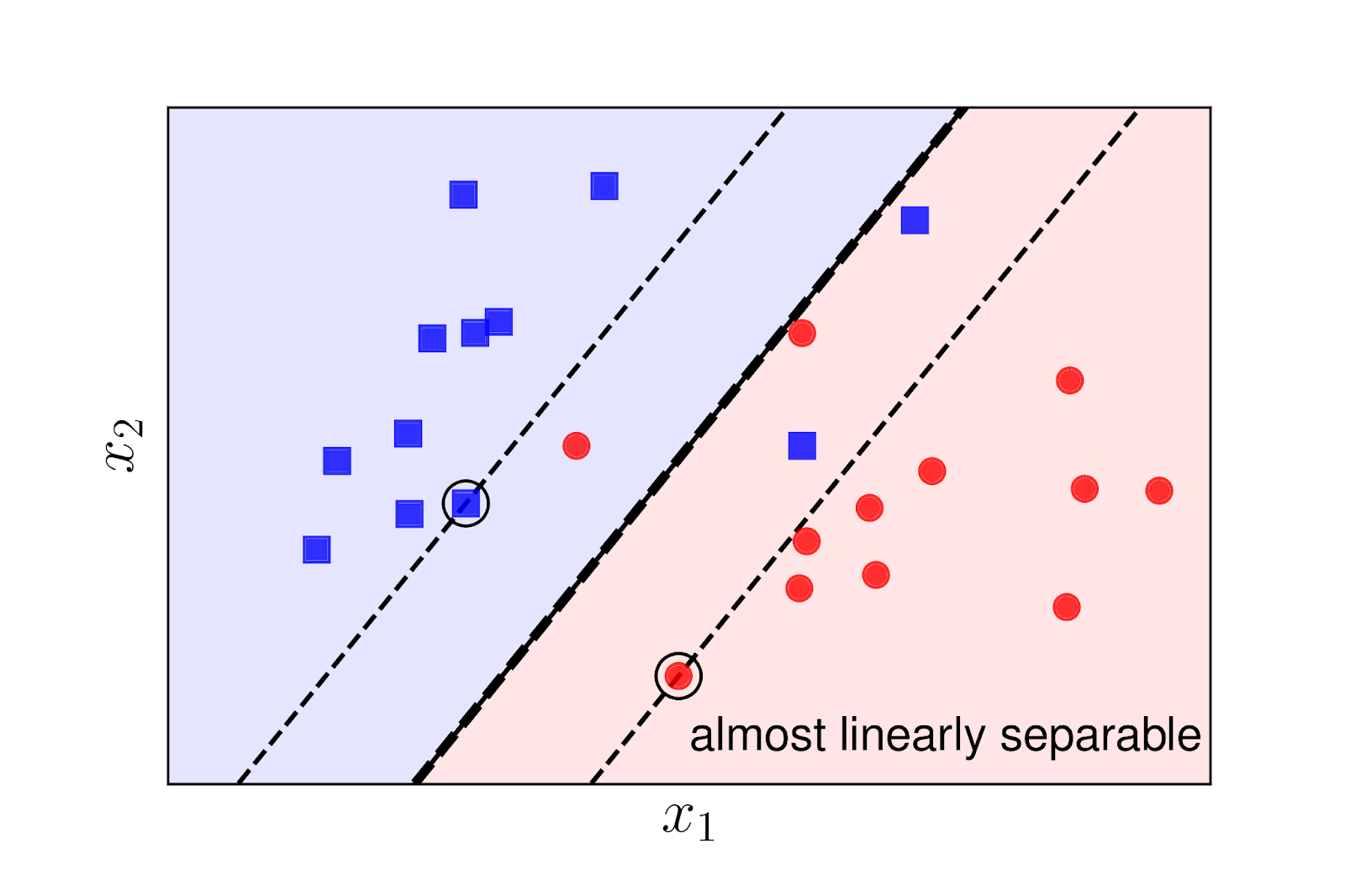
SVM là một thuật toán học có giám sát (*supervised-learning)*, nó có thể sử dụng cho cả bài toán phân lớp hoặc hồi quy. Mục tiêu của SVM là tìm ra một siêu phẳng trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Nói theo ngôn ngữ của đại số tuyến tính, siêu phẳng này phải có lề cực đại và phân chia hai bao lồi và cách đều chúng.



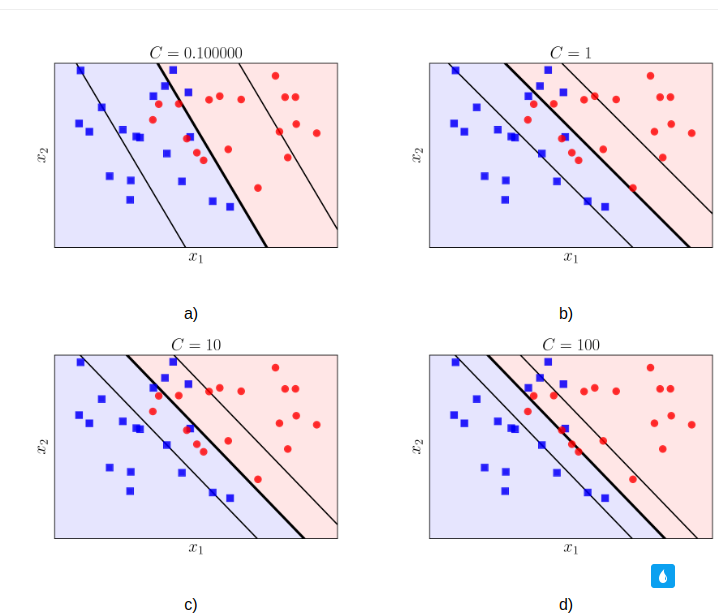
SVM là bài toán đi tìm mặt phân cách sao cho *margin* tìm được là lớn nhất trong đó *margin được hiểu là khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn* , đồng nghĩa với việc các điểm dữ liệu *an toàn nhất* so với mặt phân cách. Nếu nếu để siêu phẳng quá fit với dữ liệu sẽ làm mất tính khái quát tổng thể lúc đó SVM làm việc không hiệu quả hoặc thậm chí không làm việc



Do đó chúng ta chấp nhận lỗi xảy ra ở một vài điểm dữ liệu. Lỗi này được xác định bằng khoảng cách từ điểm đó tới đường biên tương ứng để mô hình được tổng quát hóa hơn.



ta có thể điểm chỉnh tham số C với hằng số dương giúp cân đối độ lớn của *margin* và *sự hy sinh* của các điểm nằm trong vùng *không an toàn* Khi C càng lớn thì biên càng nhỏ và ngược lại .Khi C=∞ hoặc rất lớn, Soft Margin SVM trở thành Hard Margin SVM.



### 3.2.2. Những vấn đề liên quan

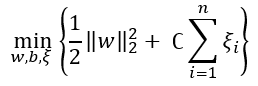
Xét bài toán phân lớp nhị phân. Cho trước một tập dữ liệu huấn luyện gồm n mẫu:



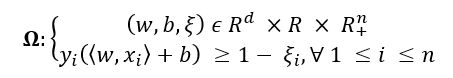
trong đó, *xi* là một véc tơ trong không gian *Rd* và *yi* ∈{-1,1} là tập các nhãn lớp. Một siêu phẳng phân tách tập *X* thành hai miền có dạng:



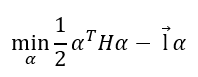
với *w* ∈ *Rd* và *b* ∈ *R*. Mục tiêu của bài toán huấn luyện SVM là tìm ra một siêu phẳng phân tách “tốt nhất” tập X theo nghĩa là lề của siêu phẳng (margin) đạt cực đại. Để tìm được bộ (*w, b*) như vậy, ta giải bài toán tối ưu sau:



sao cho thỏa mãn:



trong đó, á.,.ñ là một tích vô hướng được định nghĩa trong không gian *Rn*, *xi* là các biến *slack* được thêm vào để nới lỏng điều kiện phân lớp và *C* là tham số điều chỉnh. Thay vì giải bài toán trên, ta thường xem xét bài toán đối ngẫu của nó như sau:



trong đó thỏa mãn  . Trong đó y = (y1, y2,... yn,) , là véc tơ với toàn bộ các thành phần đều bằng l và H là một ma trận đối xứng được xác định bởi:



Ở đây, 𝞥(*.*) là một ánh xạ từ không gian ban đầu (*input space*) sang không gian đặc trưng (*feature space*) có số chiều cao hơn nhằm xử lý trường hợp dữ liệu không phân tách tuyến tính. Hàm *K* (.) được gọi là hàm nhân (*kernel function*) được định nghĩa:



# IV. Kiểm thử

## 3.1. Kết quả test perfomance

#### 

## 3.2. Đánh giá và kết luận

# Tài liệu tham khảo