**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**
**KHOA CÔNG NGHỆ THÔNG TIN**


# BÀI TIỂU LUẬN

## XỬ LÝ NGÔN NGỮ TỰ NHIÊN

**Đề tài:**

## Ứng dụng mô hình SVM cho bài toán phân loại tin tức

Sinh viên thực hiện: **Lê Văn Tiến**

Lớp: **CNTT K21CLC**

Giảng viên hướng dẫn: **TS. Trần Văn Khánh**

Thái Nguyên, năm 2025

# MỤC LỤC

# MỞ ĐẦU

Trong kỷ nguyên số, âm nhạc trực tuyến đã trở thành một phần không thể thiếu trong đời sống con người. Người dùng không chỉ nghe nhạc mà còn thường xuyên tạo và chia sẻ các danh sách phát (playlist) để phục vụ nhiều mục đích khác nhau như học tập, thư giãn, tập luyện thể thao hay giải trí cá nhân. Tuy nhiên, việc đặt tiêu đề hấp dẫn, ngắn gọn và phù hợp cho danh sách phát lại không hề đơn giản, đặc biệt khi số lượng bài hát ngày càng lớn và đa dạng về thể loại, tâm trạng cũng như nghệ sĩ. Một tiêu đề hay có thể giúp người nghe dễ dàng lựa chọn, nâng cao trải nghiệm thưởng thức, đồng thời tăng giá trị gợi ý của hệ thống âm nhạc.

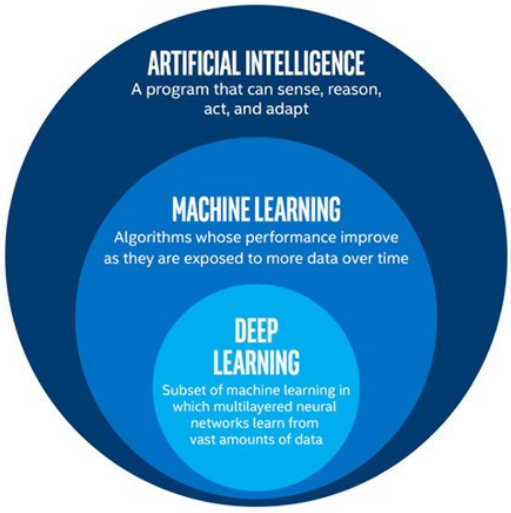
Trước nhu cầu đó, các phương pháp đặt tiêu đề thủ công hoặc dựa trên quy tắc đơn giản đã bộc lộ nhiều hạn chế: thiếu tính sáng tạo, khó thích ứng với sở thích phong phú của người dùng và không phản ánh đúng ngữ nghĩa hay cảm xúc ẩn chứa trong danh sách phát. Đây chính là lúc các mô hình học sâu hiện đại, đặc biệt là Transformer, thể hiện vai trò vượt trội. Với khả năng xử lý ngôn ngữ tự nhiên và học được các mối quan hệ ngữ cảnh phức tạp, Transformer cho phép sinh ra các tiêu đề mang tính tự nhiên, sáng tạo và phù hợp với nội dung âm nhạc. Một số nghiên cứu gần đây, như của Kim và cộng sự (2023), đã chứng minh tính khả thi của việc khai thác thông tin nghệ sĩ để tự động sinh tiêu đề playlist [**?**].

Xuất phát từ tầm quan trọng và tính thực tiễn đó, tiểu luận này tập trung vào việc nghiên cứu và ứng dụng mô hình Transformer cho bài toán sinh tiêu đề danh sách phát âm nhạc. Mục tiêu của đề tài là tìm hiểu các phương pháp tiền xử lý dữ liệu âm nhạc và văn bản, khai thác sức mạnh của Transformer để tự động sinh tiêu đề chất lượng, qua đó góp phần nâng cao trải nghiệm người dùng và mở rộng ứng dụng trí tuệ nhân tạo trong lĩnh vực giải trí số.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1 Machine learning

Machine Learning là một tập con của trí tuệ nhân tạo. Machine Learning là một lĩnh vực nhỏ trong khoa học máy tính, có khả năng tự học hỏi dựa trên dữ liệu được đưa vào mà không cần phải được lập trình cụ thể (Machine Learning is the subfield of computer science, that "gives computers the ability to learn without being explicitly programmed" – Wikipedia).




Hình 1.1. Mối quan hệ giữa học sâu, học máy và trí tuệ nhân tạo

Machine learning sử dụng các thuật toán để máy tính có thể học từ tập dữ liệu được cung cấp, từ đó, máy tính sẽ có thể thực hiện được các công việc theo yêu cầu của mỗi bài toán chi tiết dựa trên những gì đã học được. Vì vậy, máy tính có khả năng cải thiện chính nó dựa vào các tập dữ liệu mẫu được đưa vào mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Các mô hình tự học có khả năng dự đoán kết quả và phân loại thông tin mà không cần sự can thiệp của con người.

Cả thuật toán học máy sử dụng mạng lưới thần kinh để "học" từ lượng dữ liệu khổng lồ. Các mạng lưới thần kinh này là các cấu trúc có lập trình được mô phỏng theo quá trình ra quyết định của bộ não con người. Chúng bao gồm các lớp nút được kết nối với nhau để trích xuất các đặc điểm từ dữ liệu và đưa ra dự đoán về những gì dữ liệu thể hiện. Các thuật toán học máy cổ điển sử dụng mạng thần kinh với lớp đầu vào, một hoặc hai lớp 'ẩn' và lớp đầu ra.

Machine learning có mối quan hệ rất mật thiết đối với statistics (thống kê). Machine learning sử dụng các mô hình thống kê để "ghi nhớ"lại sự phân bố của dữ liệu. Tuy nhiên, không đơn thuần là ghi nhớ, machine learning phải có khả năng tổng quát hóa những gì đã được nhìn thấy và đưa ra dự đoán cho những trường hợp chưa được nhìn thấy. Đỉnh cao của machine learning sẽ là mô phỏng được khả năng tổng quát hóa và suy luận của con người để từ đó, đưa ra kết quả cho những trường hợp chưa có trong tập dữ liệu mẫu.

Từ đó, ta nhận thấy rằng khả năng phán đoán chính xác của các mô hình học máy phụ thuộc khá nhiều

và tập dữ liệu mẫu dùng để training. Từ một tập dữ liệu đơn giản, mô hình học máy đã có thể học và từ đó phán đoán một giá trị mới bất kỳ. Tuy nhiên, khi ta cho mô hình máy học một tập dữ liệu lớn hơn và chi tiết hơn về các trường hợp có thể xảy ra của dữ liệu thì chắc chắn rằng khả năng học hỏi và phán đoán kết quả của mô hình sẽ chính xác hơn rất nhiều so với tập dữ liệu nhỏ. Bên cạnh việc chuẩn bị tập dữ liệu phù hợp và đủ lớn thì một phần quan trọng không thể thiếu chính là thuật toán mà mô hình học máy đó sử dụng. Với một thuật toán tốt, có thể biểu diễn tốt hơn tập dữ liệu học thì chắc chắn rằng mô hình học máy được đào tạo ra có thể thực hiện tốt hơn nhiệm vụ phán đoán kết quả của một giá trị mới.

### 1.1.1 Phân loại machine learning

#### 1.1.1.1 Học có giám sát

Học có giám sát là một loại học máy sử dụng các tập dữ liệu được gắn nhãn để huấn luyện các thuật toán nhằm dự đoán kết quả và nhận ra các mẫu. Không giống như học không giám sát , các thuật toán học có giám sát được đào tạo có gắn nhãn để tìm hiểu mối quan hệ giữa đầu vào và đầu ra.

Dữ liệu được sử dụng trong học có giám sát được gắn nhãn - nghĩa là nó chứa các ví dụ về cả đầu vào (được gọi là đặc trưng) và đầu ra chính xác (nhãn). Các thuật toán phân tích một tập dữ liệu lớn gồm các cặp huấn luyện này để suy ra giá trị đầu ra mong muốn khi được yêu cầu đưa ra dự đoán về dữ liệu mới. Sau khi mô hình đã được huấn luyện và thử nghiệm, bạn có thể sử dụng nó để đưa ra dự đoán về dữ liệu chưa biết dựa trên kiến thức đã học trước đó.

Học có giám sát trong học máy thường được chia thành hai loại: classification (phân loại) và regression (hồi quy).

Hồi quy (Regression): Khi kết quả đầu ra là một giá trị liên tục, chúng ta sử dụng thuật toán hồi quy. Ví dụ: dự đoán giá nhà, dự đoán doanh số.

Phân loại (Classification): Khi kết quả đầu ra là một giá trị rời rạc, chúng ta sử dụng thuật toán phân loại. Ví dụ: phân loại văn bản, nhận dạng khuôn mặt.

#### 1.1.1.2 Học không giám sát

Học tập không giám sát trong trí tuệ nhân tạo là một loại học máy học từ dữ liệu mà dữ liệu đầu vào không có nhãn. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Có ba loại học máy không giám sát: phân cụm, quy tắc kết hợp và giảm kích thước.

Phân cụm (Clustering): Phân cụm là một kỹ thuật để khám phá dữ liệu thô, chưa được gắn nhãn và chia nó thành các nhóm (hoặc cụm) dựa trên những điểm tương đồng hoặc khác biệt.

Liên kết (Association): Khai thác quy tắc kết hợp là một cách tiếp cận dựa trên quy tắc để khám phá mối quan hệ thú vị giữa các điểm dữ liệu trong bộ dữ liệu lớn. Các thuật toán học không giám sát tìm kiếm các liên kết nếu – thì thường xuyên còn gọi là quy tắc để khám phá các mối tương quan và sự xuất hiện đồng thời trong dữ liệu cũng như các kết nối khác nhau giữa các đối tượng dữ liệu.

Giảm kích thước (Dimensionality reduction): Giảm kích thước là một khía cạnh thiết yếu của học máy vì nó giúp mang lại kết quả chính xác hơn cho các tập dữ liệu lớn bằng cách giúp giảm số lượng tính năng được xem xét kỹ lưỡng, giúp dữ liệu dễ quản lý hơn mà không cần loại bỏ bất kỳ phần không thể thiếu nào. Điều này giúp tránh được vấn đề về chiều và tạo ra một mô hình dự đoán phù hợp hơn.

#### 1.1.1.3 Học tăng cường

Học tăng cường (RL) là kỹ thuật máy học (ML) giúp đào tạo phần mềm đưa ra quyết định nhằm thu về kết quả tối ưu nhất. Kỹ thuật này bắt chước quy trình học thử và sai mà con người sử dụng để đạt được mục tiêu đã đặt ra. RL giúp phần mềm tăng cường các hành động hướng tới mục tiêu, đồng thời bỏ qua các hành động làm xao lãng mục tiêu.

Học tăng cường khác với học có giám sát ở chỗ trong học có giám sát, dữ liệu huấn luyện có khóa trả lời nên mô hình được huấn luyện với chính câu trả lời đúng trong khi học tăng cường, không có câu trả lời nhưng tác nhân tăng cường quyết định phải làm gì thực hiện nhiệm vụ được giao. Trong trường hợp không có tập dữ liệu huấn luyện, nó buộc phải học hỏi từ kinh nghiệm của mình.

### 1.1.2 Một số thuật toán học máy cơ bản

#### 1.1.2.1 Hồi quy

Hồi quy, một phương pháp thống kê, phân tích mối quan hệ giữa các biến phụ thuộc và biến độc lập, cho phép dự đoán thông qua các mô hình hồi quy khác nhau.

Regression hay Hồi quy là một phương pháp thống kê được sử dụng để phân tích mối quan hệ giữa một biến phụ thuộc (biến mục tiêu) và một hoặc nhiều biến độc lập (biến dự đoán). Mục tiêu là xác định hàm phù hợp nhất mô tả mối liên hệ giữa các biến này.

Đây là một kỹ thuật học máy có giám sát, được sử dụng để dự đoán giá trị của biến phụ thuộc cho dữ liệu mới, chưa được nhìn thấy. Nó mô hình hóa mối quan hệ giữa các tính năng đầu vào và biến mục tiêu, cho phép ước tính hoặc dự đoán các giá trị số.

**a) Linear Regression**

Hồi quy tuyến tính là một trong những mô hình thống kê đơn giản và được sử dụng rộng rãi nhất. Điều này giả định rằng có mối quan hệ tuyến tính giữa các biến độc lập và biến phụ thuộc. Điều này có nghĩa là sự thay đổi của biến phụ thuộc tỷ lệ thuận với sự thay đổi của biến độc lập.

**b) Logistic Regression**

Hồi quy Logistic là một mô hình thống kê được sử dụng để phân loại nhị phân, tức dự đoán một đối tượng thuộc vào một trong hai nhóm. Hồi quy Logistic làm việc dựa trên nguyên tắc của hàm sigmoid – một hàm phi tuyến tự chuyển đầu vào của nó thành xác suất thuộc về một trong hai lớp nhị phân.

**c) Stepwise regression**

Phương pháp hồi quy từng bước (stepwise regression) là một phương pháp phân tích hồi quy được sử dụng để xác định các biến độc lập quan trọng nhất trong việc dự đoán biến phụ thuộc. Phương pháp này tiến hành bổ sung các biến một cách tuần tự vào mô hình hồi quy, đánh giá ảnh hưởng và khả năng giải thích của từng biến được bổ sung. Quá trình hồi quy từng bước bắt đầu với một mô hình hồi quy đơn giản chứa một biến độc lập duy nhất. Sau đó, các biến khác được bổ sung vào mô hình dựa trên các tiêu chí nhất định, như mức độ quan trọng của biến đối với biến phụ thuộc, sự cải thiện của mô hình sau khi thêm biến, và các chỉ số thống kê như giá trị p. Phương pháp hồi quy từng bước (stepwise regression) là một kỹ thuật thống kê được sử dụng để xây dựng một mô hình hồi quy tuyến tính bằng cách chọn ra tập hợp các biến độc lập quan trọng nhất để dự đoán biến phụ thuộc. Phương pháp này thường được áp dụng trong việc lựa chọn biến trong mô hình hồi quy khi có nhiều biến độc lập có thể ảnh hưởng đến biến phụ thuộc.

Quá trình tiến hành theo hai hướng: tiến và lùi. Trong hướng tiến, một biến độc lập được bổ sung vào mô hình ở mỗi bước và kiểm tra xem biến đó có cải thiện khả năng giải thích của mô hình hay không. Nếu biến đó đạt được ngưỡng quy định, nó sẽ được giữ lại trong mô hình. Trong hướng lùi, các biến độc lập được loại bỏ khỏi mô hình một cách tuần tự để kiểm tra xem loại bỏ biến đó có làm giảm khả năng giải thích của mô hình hay không. Phương pháp hồi quy từng bước cho phép chúng ta tìm ra một mô hình hồi quy tối ưu, chỉ chứa các biến quan trọng nhất và đóng góp đáng kể vào việc dự đoán biến phụ thuộc. Nó giúp giảm chiều của mô hình và loại bỏ các biến không cần thiết, từ đó tăng tính hiệu quả và khả năng giải thích của mô hình hồi quy.

Tuy nhiên, cần lưu ý rằng phương pháp hồi quy từng bước cũng có nhược điểm, bao gồm khả năng tạo ra mô hình quá đơn giản hoặc quá phức tạp, vấn đề về đa cộng tuyến (multicollinearity) giữa các biến độc lập, và nguy cơ xảy ra sai sót thống kê. Do đó, việc áp dụng phương pháp này cần cân nhắc kỹ lưỡng và kết hợp với các phương pháp khác để đánh giá mô hình hồi quy một cách toàn diện.

#### 1.1.2.2 Classification

Classification là một quá trình phân loại dữ liệu hoặc đối tượng thành các lớp hoặc danh mục được xác định trước dựa trên các tính năng hoặc thuộc tính của chúng.

Phân loại là một loại kỹ thuật học có giám sát trong đó thuật toán được đào tạo trên tập dữ liệu được gắn nhãn để dự đoán lớp hoặc danh mục dữ liệu mới, chưa được nhìn thấy.

Mục tiêu chính của học máy phân loại là xây dựng một mô hình có thể gán nhãn hoặc danh mục chính xác cho một quan sát mới dựa trên các tính năng của nó.

**a) Linear Classification.**

Linear Classifier là một loại mô hình học máy được sử dụng trong bài toán phân loại. Mô hình này

hoạt động dựa trên nguyên tắc của hàm tuyến tính để tạo ra một ranh giới phẳng (hyperplane) trong không gian đặc trưng, phân chia các điểm dữ liệu thành các lớp khác nhau.

Cụ thể, trong một không gian đặc trưng nhiều chiều, mỗi điểm dữ liệu được biểu diễn dưới dạng một vector đặc trưng. Một linear classifier sẽ tính toán một tổ hợp tuyến tính của các đặc trưng này và sử dụng kết quả để quyết định lớp mà điểm dữ liệu thuộc về. Thường thì, một hàm kích hoạt (ví dụ: hàm softmax cho bài toán phân loại nhiều lớp hoặc hàm sigmoid cho bài toán phân loại nhị phân) được sử dụng để chuyển đổi đầu ra của tổ hợp tuyến tính thành xác suất thuộc về từng lớp.

**b) K-Nearest Neighbor(KNN)**

Kernel Support Vector Machine (Kernel SVM) là một biến thể của Support Vector Machine (SVM) được sử dụng để giải quyết các bài toán phân loại không thể phân chia tuyến tính. Trong SVM, mục tiêu là tìm ra một ranh giới phẳng (hyperplane) phân chia các lớp sao cho khoảng cách từ các điểm dữ liệu gần nhất đến hyperplane là lớn nhất có thể. Tuy nhiên, đôi khi các lớp dữ liệu không thể được phân chia hoàn toàn bằng một hyperplane tuyến tính.

Kernel SVM giải quyết vấn đề này bằng cách sử dụng một kỹ thuật gọi là "kernel trick". Thay vì phân loại trực tiếp trong không gian đặc trưng ban đầu, kernel SVM ánh xạ dữ liệu vào một không gian đặc trưng cao hơn (thường là không gian nhiều chiều hơn) thông qua một hàm kernel. Trong không gian đặc trưng mới này, dữ liệu có thể được phân chia tuyến tính bằng một hyperplane.

**c) Support Vector Machine (SVM)**

SVM là một thuật toán học máy có giám sát được sử dụng cho cả phân loại và hồi quy. Mặt phẳng quyết định (siêu phẳng) là mặt phẳng phân tách giữa một tập hợp các đối tượng có các thành viên lớp khác nhau. Mục tiêu chính của thuật toán SVM là tìm siêu phẳng tối ưu trong không gian N chiều có thể phân tách các điểm dữ liệu trong các lớp khác nhau trong không gian đặc trưng. Siêu phẳng cố gắng sao cho khoảng cách giữa các điểm gần nhất của các lớp khác nhau phải lớn nhất có thể. Một lựa chọn hợp lý được coi là siêu phẳng tốt nhất là siêu phẳng thể hiện khoảng cách hoặc lề lớn nhất giữa hai lớp.

Ngoài ra còn các thuật toán khác như:

-Instance-based Algorithms

-Regularization Algorithms

-Bayesian Algorithms

-Clustering Algorithms

-Artificial Neural Network Algorithms

-Dimensionality Reduction Algorithms

-Ensemble Algorithms

## 1.2 Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) là một nhánh quan trọng của Trí tuệ nhân tạo (Artificial Intelligence – AI), tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người. Ngôn ngữ tự nhiên vốn là công cụ giao tiếp cơ bản nhất của con người, chứa đựng sự phong phú, đa dạng và phức tạp trong cách diễn đạt, vì vậy việc giúp máy tính có khả năng hiểu, phân tích và sinh ngôn ngữ một cách tự nhiên như con người là một trong những mục tiêu đầy tham vọng của ngành khoa học máy tính. Mục tiêu cốt lõi của NLP là giúp máy tính hiểu, diễn giải và thực hiện hiệu quả những nhiệm vụ liên quan đến ngôn ngữ, bao gồm tương tác giữa người và máy, cải thiện hiệu quả giao tiếp giữa con người với con người thông qua công nghệ trung gian, cũng như nâng cao hiệu quả xử lý văn bản và lời nói trong các hệ thống thông minh. Thực tế, một trợ lý ảo có thể "nghe" yêu cầu của người dùng bằng giọng nói, sau đó phân tích, hiểu ý nghĩa và phản hồi lại bằng lời nói tự nhiên, điều này mang lại trải nghiệm gần gũi hơn, giúp công nghệ trở nên hữu ích và thân thiện trong đời sống hàng ngày. Xét về lịch sử, NLP được nghiên cứu từ giữa thế kỷ 20 khi các nhà khoa học nghĩ tới việc dịch tự động ngôn ngữ, nhưng thời kỳ đầu chủ yếu dựa vào luật do con người xây dựng thủ công. Cách tiếp cận dựa trên quy tắc này sớm bộc lộ hạn chế vì ngôn ngữ luôn biến đổi, khó bao quát bằng luật cứng nhắc. Bước ngoặt chỉ đến khi học máy và đặc biệt là học sâu xuất hiện, cho phép hệ thống học trực tiếp từ dữ liệu, thay vì phụ thuộc vào quy tắc cố định. Sự ra đời của kiến trúc Transformer năm 2017 đã mở ra kỷ nguyên mới cho NLP, khi các mô hình ngôn ngữ lớn (Large Language Models – LLMs) ra đời, có khả năng hiểu và sinh văn bản tự nhiên gần giống con người. Tuy nhiên, NLP cũng đối diện với nhiều thách thức. Một trong số đó là sự đa nghĩa và mơ hồ: một từ hay một câu có thể mang nhiều nghĩa khác nhau tùy ngữ cảnh, ví dụ từ "bank" có thể là "ngân hàng" hoặc "bờ sông". Bên cạnh đó, ngữ cảnh và yếu tố văn hóa ảnh hưởng mạnh đến cách hiểu ngôn ngữ, khiến máy tính khó xử lý chính xác. Cấu trúc ngôn ngữ cũng rất phức tạp, câu văn có thể đảo ngữ, rút gọn hoặc dùng ẩn dụ, làm tăng thêm độ khó. Đặc biệt, nhiều ngôn ngữ ít tài nguyên như tiếng Việt vẫn gặp khó khăn trong việc phát triển NLP do hạn chế dữ liệu. Dù vậy, NLP ngày nay đã đạt được nhiều ứng dụng rộng rãi trong đời sống. Các công cụ tìm kiếm như Google Search sử dụng NLP để hiểu ý định truy vấn; dịch máy như Google Translate hay DeepL cho phép dịch hàng trăm ngôn ngữ; trợ lý ảo và chatbot như Siri, Alexa, Google Assistant hay ChatGPT có khả năng trò chuyện tự nhiên, hỗ trợ công việc và giải trí; các hệ thống phân tích cảm xúc giúp doanh nghiệp hiểu rõ hơn phản hồi khách hàng; trong lĩnh vực y học, NLP được dùng để phân tích tài liệu, hỗ trợ chẩn đoán, tóm tắt nghiên cứu khoa học; đồng thời trong hành chính – văn phòng, NLP giúp tự động phân loại, trích xuất thông tin từ văn bản, tiết kiệm nhiều công sức cho con người. Trong số các mô hình NLP nổi bật hiện nay có thể kể đến ChatGPT của OpenAI – mô hình hội thoại thông minh được ứng dụng rộng rãi, Gemini của Google DeepMind – chú trọng vào tích hợp đa phương thức, Poe của Quora – nền tảng tập hợp nhiều chatbot AI, cùng các trợ lý ảo quen thuộc như Siri, Alexa hay Google Assistant. Có thể thấy, NLP không chỉ dừng lại ở lĩnh vực nghiên cứu mà đã trở thành công nghệ thực tiễn, thay đổi cách con người tương tác với máy tính và với nhau. Với sự phát triển nhanh chóng của trí tuệ nhân

tạo, NLP hứa hẹn sẽ tiếp tục mở rộng, tạo ra nhiều ứng dụng mới và mang lại bước tiến lớn trong việc kết nối con người với công nghệ.

### 1.2.1 Một số bài toán trong NLP

Xử lý ngôn ngữ tự nhiên được ứng dụng trong nhiều bài toán thực tế, cụ thể như sau:

- Mô hình hóa ngôn ngữ (Language modelling)

Mô hình hóa ngôn ngữ (LM) gán một xác suất cho bất kỳ chuỗi từ nào. Về cơ bản, trong bài toán này, ta cần dự đoán từ tiếp theo xuất hiện theo trình tự, dựa trên lịch sử của các từ đã xuất hiện trước đó. LM rất quan trọng trong các ứng dụng khác nhau của NLP, và là lý do tại sao máy móc có thể hiểu được thông tin định tính. Một số ứng dụng của Mô hình hóa ngôn ngữ bao gồm: nhận dạng giọng nói, nhận dạng ký tự quang học, nhận dạng chữ viết tay, dịch máy và sửa lỗi chính tả.

- Phân loại văn bản (Text classification)

Phân loại văn bản gán các danh mục được xác định trước cho văn bản dựa trên nội dung của nó. Cho đến nay, phân loại văn bản là ứng dụng phổ biến nhất của NLP, được sử dụng để xây dựng các công cụ khác nhau như trình phát hiện thư rác và chương trình phân tích cảm xúc.

- Trích xuất thông tin (Information extraction)

Trích xuất thông tin (IE) tự động trích xuất thông tin có liên quan từ các tài liệu văn bản không có cấu trúc và / hoặc bán cấu trúc. Ví dụ về các loại tài liệu này bao gồm lịch sự kiện từ email hoặc tên của những người được đề cập trong một bài đăng trên mạng xã hội.

- Truy xuất thông tin (Information retrieval)

Trích xuất thông tin là bài toán làm nhiệm vụ tìm kiếm các tài liệu có liên quan từ một bộ dữ liệu lớn các tài liệu liên quan đến truy vấn do người dùng thực hiện. Google là một loại hệ thống Truy xuất Thông tin (IR) phổ biến nhất mà chúng ta thường sử dụng.

- Tác tử phần mềm hội thoại (Conversational agent)

Tác tử phần mềm hội thoại thuộc AI hội thoại, liên quan đến việc xây dựng các hệ thống đối thoại mô phỏng các tương tác của con người. Các ví dụ phổ biến về AI hội thoại bao gồm Alexa, Siri, Google Home, Cortana, hay trợ lý ảo ViVi. Các công nghệ như chatbot cũng được hỗ trợ bởi tác tử phần mềm hội thoại và ngày càng phổ biến trong các doanh nghiệp.

- Tóm tắt văn bản (Text summarization)

Tóm tắt văn bản là quá trình rút ngắn một tập hợp dữ liệu để tạo một tập hợp con đại diện cho thông tin quan trọng nhất hoặc có liên quan trong nội dung gốc.

- Hỏi đáp (Question answering)

Hỏi đáp là bài toán xây dựng các hệ thống có thể tự động trả lời cho các câu hỏi do con người đặt ra bằng ngôn ngữ tự nhiên. Đây là bài toán có thể coi là tổng quan nhất trong NLP, nó có thể thực hiện được nhiều nhiệm vụ khác nhau trong NLP: tóm tắt văn bản, hỏi đáp, truy xuất nội dung, phân loại văn bản...

- Dịch máy (Machine translation)

Dịch máy (MT) là một nhánh con của ngôn ngữ học tính toán liên quan đến việc chuyển đổi một đoạn văn bản từ ngôn ngữ này sang ngôn ngữ khác. Một ứng dụng phổ biến của loại này là Google Dịch.

- Mô hình hóa chủ đề (Topic modelling)

Mô hình hóa chủ đề là một kỹ thuật Học máy không giám sát giúp khám phá cấu trúc chủ đề của một bộ tài liệu lớn. Ứng dụng NLP này là một công cụ khá phổ biến, được sử dụng trên nhiều lĩnh vực khác nhau – như Văn học, và Tin sinh học.

### 1.2.2 Một số mô hình trong NLP

Trong xử lý ngôn ngữ tự nhiên (NLP), có nhiều mô hình và kiến trúc khác nhau được sử dụng để giải quyết các tác vụ như dịch máy, phân loại văn bản, nhận diện thực thể, tóm tắt văn bản,...

#### 1.2.2.1 Mô hình túi từ - Bag of word(BoW)

Mô hình **Túi từ (Bag of Words – BoW)** là một trong những phương pháp cơ bản và lâu đời nhất để biểu diễn văn bản trong xử lý ngôn ngữ tự nhiên. Ý tưởng chính của BoW là coi mỗi văn bản như một "túi" chứa các từ, trong đó chỉ quan tâm đến sự xuất hiện và tần suất của từ, mà bỏ qua thứ tự, ngữ pháp hay ngữ nghĩa. Với cách tiếp cận này, mỗi tài liệu được biểu diễn dưới dạng một vector trong không gian đặc trưng, trong đó mỗi chiều tương ứng với một từ trong tập từ vựng chung.

Giả sử tập tài liệu $D$ có từ vựng $V = \{w_1, w_2, \ldots, w_{|V|}\}$. Mỗi tài liệu $d \in D$ có thể được biểu diễn bằng vector:

$$\mathbf{v}_d = \big(f(w_1,d), f(w_2,d), \ldots, f(w_{|V|},d)\big)$$

trong đó $f(w_i,d)$ là số lần xuất hiện (tần suất) của từ $w_i$ trong tài liệu $d$.

Ví dụ, nếu tập từ vựng gồm ba từ *"tốt"*, *"xấu"*, *"bình thường"*, thì câu *"sản phẩm này tốt và rất tốt"* có thể được biểu diễn bằng vector $[2,0,0]$, trong đó số 2 thể hiện từ *"tốt"* xuất hiện hai lần trong câu.

Mặc dù đơn giản, mô hình BoW vẫn tỏ ra hiệu quả trong nhiều bài toán thực tế, đặc biệt khi kết hợp với các thuật toán học máy truyền thống như **Naive Bayes**, **Logistic Regression**, hay **SVM (Support Vector Machine)**. Điểm mạnh của BoW nằm ở tính dễ triển khai, trực quan và khả năng xử lý nhanh trên dữ liệu có kích thước vừa và nhỏ. BoW thường được sử dụng trong những tác vụ không yêu cầu phân tích cú pháp phức tạp như **phân tích cảm xúc (Sentiment Analysis)**, **phân loại văn bản (Text Classification)**, **phát hiện thư rác (Spam Detection)** hay các hệ thống gợi ý dựa trên mô tả ngắn gọn.

Tuy nhiên, BoW cũng tồn tại nhiều hạn chế. Do không xét đến **thứ tự từ** và **ngữ nghĩa**, mô hình có thể bỏ qua nhiều thông tin quan trọng của văn bản. Hai câu có nghĩa khác nhau nhưng cùng chứa các từ giống nhau có thể bị biểu diễn giống hệt nhau. Ngoài ra, số chiều của vector BoW thường rất lớn vì phải bao quát toàn bộ tập từ vựng, dẫn đến vấn đề **thưa thớt dữ liệu (sparsity)** và tốn tài nguyên tính toán. Chính vì vậy,

BoW thường được coi là bước khởi đầu cơ bản, và đã thúc đẩy sự phát triển của các phương pháp tiên tiến hơn như **TF–IDF**, **Word2Vec**, **GloVe**, hay **BERT**.

#### 1.2.2.2 Mô hình TF - IDF

Mô hình **TF–IDF (Term Frequency – Inverse Document Frequency)** là một cải tiến quan trọng của mô hình **Bag of Words (BoW)**, nhằm khắc phục hạn chế của BoW trong việc đánh giá tầm quan trọng của các từ. Nếu như BoW chỉ đơn thuần đếm số lần xuất hiện của từ trong văn bản, dẫn đến việc các từ phổ biến như "và", "là", "ở" hay "của" có tần suất cao nhưng không thực sự mang nhiều giá trị về mặt ngữ nghĩa, thì TF–IDF bổ sung thêm một cơ chế trọng số để phản ánh mức độ quan trọng của từ trong mối tương quan giữa văn bản hiện tại và toàn bộ tập tài liệu.

Cụ thể, **TF (Term Frequency)** đo lường tần suất xuất hiện của một từ $t$ trong một tài liệu $d$, được tính theo công thức:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

trong đó $f_{t,d}$ là số lần xuất hiện của từ $t$ trong tài liệu $d$.

**IDF (Inverse Document Frequency)** phản ánh độ hiếm của từ trong toàn bộ tập tài liệu $D$, được định nghĩa như sau:

$$IDF(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

trong đó $N$ là tổng số tài liệu trong tập $D$, và $|\{d \in D : t \in d\}|$ là số lượng tài liệu chứa từ $t$.

Từ đó, trọng số TF–IDF của một từ $t$ trong tài liệu $d$ được xác định bằng:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

Theo công thức này, một từ sẽ có trọng số cao nếu nó xuất hiện nhiều lần trong tài liệu đang xét nhưng lại ít xuất hiện trong các tài liệu khác. Điều này cho phép TF–IDF nhấn mạnh những từ mang tính đặc trưng cho tài liệu, đồng thời giảm ảnh hưởng của những từ phổ biến nhưng ít giá trị thông tin.

Mô hình TF–IDF được ứng dụng rộng rãi trong nhiều bài toán xử lý ngôn ngữ tự nhiên. Một trong những ứng dụng phổ biến nhất là trong **tìm kiếm thông tin (Information Retrieval)**, khi hệ thống cần xác định mức độ liên quan giữa một truy vấn và các tài liệu trong tập dữ liệu. Ngoài ra, TF–IDF còn được dùng trong **phân loại văn bản (Text Classification)**, **phân tích cảm xúc (Sentiment Analysis)**, **lọc thư rác (Spam Filtering)**, và **tóm tắt văn bản (Text Summarization)**. Điểm mạnh của TF–IDF là đơn giản, dễ triển khai và vẫn đạt hiệu quả cao trong nhiều tác vụ thực tế, đặc biệt khi dữ liệu có kích thước vừa phải.

Tuy nhiên, TF–IDF cũng tồn tại một số hạn chế. Mô hình này vẫn chưa nắm bắt được ngữ cảnh hoặc mối quan hệ ngữ nghĩa giữa các từ, vì nó chỉ dựa trên tần suất và sự phân bố từ trong tài liệu. Do đó, hai từ đồng nghĩa hoặc các cụm từ có quan hệ ngữ nghĩa chặt chẽ có thể không được xem xét một cách chính xác. Bên cạnh

đó, TF–IDF tạo ra vector có số chiều lớn bằng kích thước từ vựng, dẫn đến vấn đề dữ liệu thưa thớt tương tự như BoW. Mặc dù vậy, nhờ tính đơn giản và hiệu quả, TF–IDF vẫn được coi là một trong những phương pháp nền tảng quan trọng trong xử lý văn bản, và thường được sử dụng như một bước tiền xử lý hoặc làm baseline để so sánh với các mô hình phức tạp hơn như Word2Vec, GloVe hay BERT.

#### 1.2.2.3 Mô hình RNN

Recurrent Neural Network (RNN) là một loại mô hình mạng nơ-ron sâu, đặc biệt hữu ích trong việc xử lý các dữ liệu dạng chuỗi, chẳng hạn như văn bản, âm thanh, video, hoặc dữ liệu thời gian thực. RNN khác với mạng nơ-ron truyền thống (như feedforward neural network) ở chỗ nó có khả năng ghi nhớ thông tin từ quá khứ, giúp mô hình có thể hiểu và xử lý dữ liệu phụ thuộc vào ngữ cảnh hoặc thứ tự. Tuy nhiên, RNN có thể gặp phải vấn đề quên lãng dần (vanishing gradient problem) khi xử lý chuỗi dài. Đặc điểm của RNN

- Kết nối tuần hoàn (Recurrent Connections):

+ Ở mỗi thời điểm trong chuỗi (ví dụ: mỗi từ trong câu), một RNN nhận một đầu vào và lưu giữ thông tin từ thời điểm trước đó thông qua trạng thái ẩn (hidden state). Điều này tạo ra một vòng lặp tuần hoàn bên trong mạng.

Kí hiệu:

$$h_t = f(h_{t-1}, x_t)$$

Trong đó:

- $h_t$ là trạng thái ẩn ở thời điểm $t$;
- $h_{t-1}$ là trạng thái ẩn ở thời điểm trước đó;
- $x_t$ là đầu vào ở thời điểm hiện tại;
- $f$ là hàm kích hoạt (thường là sigmoid hoặc tanh).

- Khả năng xử lý dữ liệu tuần tự:

RNN có khả năng ghi nhớ thông tin từ các bước trước đó trong chuỗi và sử dụng nó để xử lý đầu vào hiện tại. Điều này làm cho RNN trở thành lựa chọn phổ biến trong các bài toán liên quan đến chuỗi dữ liệu như ngôn ngữ tự nhiên, âm thanh, hay video.

- Chia sẻ trọng số:

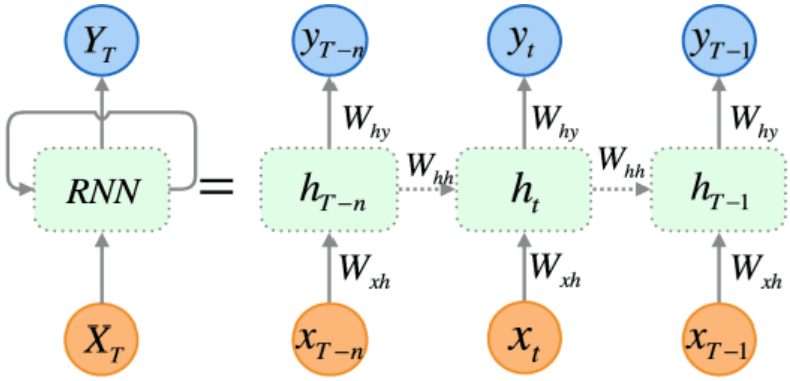
Trọng số được chia sẻ qua tất cả các bước trong chuỗi, tức là cùng một bộ trọng số được áp dụng cho mọi bước trong chuỗi đầu vào. Điều này giúp giảm số lượng tham số cần học, làm cho RNN hiệu quả khi xử lý dữ liệu chuỗi dài.

Với cơ chế trạng thái ẩn, RNN có khả năng nắm bắt và lưu giữ thông tin từ các bước trước đó trong chuỗi. Nhờ đó, đầu ra tại thời điểm hiện tại không chỉ phụ thuộc vào đầu vào hiện tại mà còn chịu ảnh hưởng bởi toàn bộ lịch sử của chuỗi. Điều này làm cho RNN đặc biệt phù hợp trong các bài toán như dịch máy (machine

translation), nhận diện giọng nói (speech recognition), phân tích chuỗi thời gian (time series analysis), hay xử lý video.

Một ưu điểm quan trọng khác của RNN là cơ chế chia sẻ trọng số. Cùng một tập trọng số $W_{hx}, W_{hh}, W_{hy}$ được sử dụng lặp lại cho tất cả các bước trong chuỗi. Nhờ đó, số lượng tham số cần học không phụ thuộc trực tiếp vào độ dài chuỗi đầu vào, đồng thời mô hình có khả năng tổng quát hóa tốt hơn và hiệu quả tính toán cao hơn.

Tuy nhiên, trong quá trình huấn luyện bằng phương pháp lan truyền ngược theo thời gian (Backpropagation Through Time - BPTT), RNN thường gặp phải hiện tượng *vanishing gradient* (gradient suy giảm) hoặc *exploding gradient* (gradient bùng nổ) khi chuỗi dữ liệu quá dài. Điều này khiến mô hình gặp khó khăn trong việc ghi nhớ các phụ thuộc dài hạn và chỉ hoạt động tốt với ngữ cảnh ngắn. Để khắc phục, các biến thể như **LSTM** (Long Short-Term Memory) và **GRU** (Gated Recurrent Unit) đã được phát triển, giúp cải thiện đáng kể hiệu quả của mạng RNN trong thực tế.




Hình 1.2: Ví dụ minh họa về mô hình RNN

#### 1.2.2.4 Mô hình LSTM

Một trong những cải tiến quan trọng của mô hình RNN cơ bản được áp dụng nhiều trong xử lý ngôn ngữ tự nhiên (NLP) là **mô hình Long Short-Term Memory (LSTM)**. Mô hình này được đề xuất bởi Hochreiter và Schmidhuber (1997) nhằm giải quyết hạn chế của RNN cơ bản trong việc ghi nhớ thông tin dài hạn, đặc biệt là vấn đề *tiêu biến gradient* (vanishing gradient) và *bùng nổ gradient* (exploding gradient) trong quá trình huấn luyện. Nhờ cấu trúc đặc biệt, LSTM có khả năng duy trì và điều chỉnh thông tin qua nhiều bước thời gian, từ đó học được các phụ thuộc dài hạn trong dữ liệu chuỗi. Hiện nay, LSTM được ứng dụng rộng rãi trong nhiều lĩnh vực như xử lý văn bản, nhận dạng giọng nói, phân tích chuỗi thời gian, và phân tích video.

Khác với RNN cơ bản chỉ có một cơ chế lặp đơn giản, LSTM sử dụng một cấu trúc phức tạp hơn nhờ việc bổ sung **các cổng (gates)** để kiểm soát luồng thông tin đi qua các trạng thái. Các cổng này hoạt động như những "van" điều chỉnh, giúp mô hình quyết định thông tin nào nên lưu giữ, thông tin nào cần bỏ qua, và thông tin nào được xuất ra tại mỗi bước thời gian. Cụ thể, một đơn vị LSTM bao gồm ba loại cổng chính: **cổng quên (Forget Gate)**, **cổng đầu vào (Input Gate)** và **cổng đầu ra (Output Gate)**.

- **Cổng quên (Forget Gate)**: Quyết định loại bỏ thông tin nào khỏi trạng thái bộ nhớ. Đây là thành phần quan trọng để tránh việc mô hình lưu trữ quá nhiều thông tin không cần thiết. Cổng này được tính theo công thức:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

trong đó $h_{t-1}$ là trạng thái ẩn tại thời điểm $t-1$, $x_t$ là đầu vào tại thời điểm $t$, $W_f$ và $b_f$ lần lượt là ma trận trọng số và vector bias. Hàm sigmoid $\sigma$ đảm bảo giá trị của $f_t$ nằm trong khoảng $(0,1)$, với 0 tương ứng loại bỏ hoàn toàn và 1 tương ứng giữ lại toàn bộ thông tin.

- **Cổng đầu vào (Input Gate)**: Xác định phần nào của thông tin mới sẽ được đưa vào bộ nhớ. Cổng này hoạt động cùng với một vector trạng thái ứng viên $\tilde{C}_t$ để cập nhật thông tin mới:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

trong đó $i_t$ là giá trị của cổng đầu vào, $\tilde{C}_t$ là thông tin ứng viên mới, và $C_t$ là trạng thái bộ nhớ tại thời điểm $t$ được cập nhật từ trạng thái cũ $C_{t-1}$.

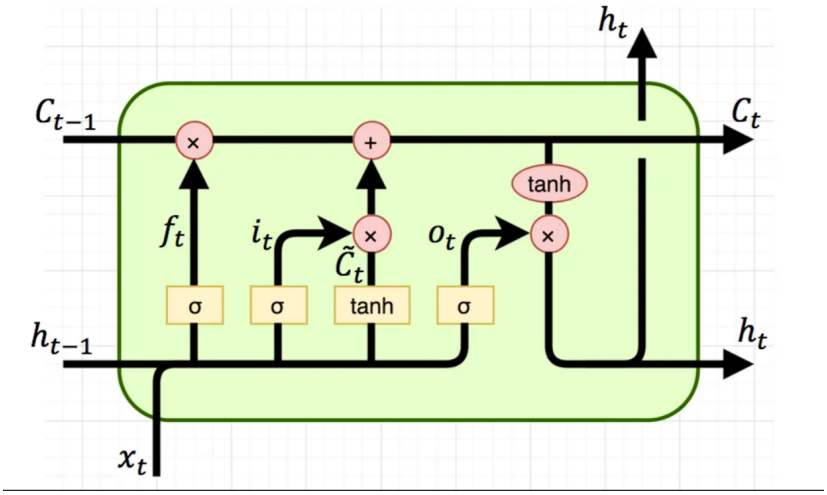
- **Cổng đầu ra (Output Gate)**: Quyết định phần nào của thông tin trong bộ nhớ sẽ được sử dụng để sinh ra trạng thái ẩn mới $h_t$. Công thức của cổng đầu ra là:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Trong đó $h_t$ chính là trạng thái ẩn tại thời điểm $t$, được dùng để dự đoán hoặc làm đầu vào cho bước tiếp theo.

Tóm lại, nhờ cơ chế ba cổng trên, LSTM có khả năng lưu giữ thông tin quan trọng qua nhiều bước thời gian và loại bỏ các thông tin không cần thiết. Điều này giúp mô hình vượt trội hơn RNN cơ bản trong việc học các phụ thuộc dài hạn. Nhờ tính hiệu quả và khả năng khắc phục nhược điểm của RNN, LSTM đã trở thành một trong những mô hình nền tảng quan trọng trong xử lý dữ liệu chuỗi, và đến nay vẫn được sử dụng rộng rãi trong nhiều ứng dụng NLP và các lĩnh vực khác.




Hình 1.4: Ví dụ minh họa về mô hình LSTM

#### 1.2.2.5 Mô hình BERT

**BERT (Bidirectional Encoder Representations from Transformers)** là một trong những mô hình quan trọng nhất trong sự phát triển của xử lý ngôn ngữ tự nhiên (NLP), được nhóm nghiên cứu của Google công bố vào năm 2018. BERT dựa trên kiến trúc Transformer, vốn bao gồm hai thành phần chính là **Encoder** và **Decoder**, nhưng chỉ sử dụng phần Encoder để xây dựng biểu diễn ngữ nghĩa của văn bản. Điểm đột phá của BERT nằm ở cơ chế **huấn luyện hai chiều (bidirectional)**, cho phép mô hình nắm bắt ngữ cảnh của từ dựa trên cả thông tin phía trước và phía sau, thay vì chỉ một chiều như GPT (từ trái sang phải) hay mô hình ngôn ngữ truyền thống.

BERT được xây dựng từ nhiều lớp Encoder của Transformer. Mỗi Encoder bao gồm hai thành phần chính: **Multi-Head Self-Attention** và **Feed-Forward Network**. Cơ chế Self-Attention cho phép mô hình tính toán mối quan hệ giữa các từ trong cùng một câu.

Cụ thể, với đầu vào là một chuỗi embedding $X = (x_1, x_2, \ldots, x_n)$, ta xây dựng ba ma trận **Query** ($Q$), **Key** ($K$) và **Value** ($V$):

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

trong đó $W_Q, W_K, W_V$ là các ma trận trọng số học được. Cơ chế Self-Attention được định nghĩa bởi công thức:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

trong đó $d_k$ là kích thước của vector Key, nhằm đảm bảo giá trị chuẩn hóa ổn định. Multi-Head Attention thực hiện nhiều phép Attention song song, sau đó gộp kết quả lại để mô hình học được nhiều loại quan hệ ngữ nghĩa khác nhau.

Trong quá trình tiền huấn luyện (pre-training), BERT kết hợp hai nhiệm vụ chính:

- **Masked Language Model (MLM)**: Thay vì dự đoán từ tiếp theo, BERT chọn ngẫu nhiên một số từ trong chuỗi đầu vào và thay thế bằng token đặc biệt [MASK]. Nhiệm vụ của mô hình là dự đoán các từ gốc dựa trên ngữ cảnh hai chiều.

  Cho một chuỗi đầu vào $X = (x_1, x_2, \ldots, x_n)$, giả sử tập $M \subset \{1, 2, \ldots, n\}$ chứa các vị trí bị che, hàm mất mát của MLM được định nghĩa là:

$$\mathscr{L}_{MLM} = -\sum_{i \in M} \log P(x_i \mid X_{\setminus M}; \theta)$$

  trong đó $X_{\setminus M}$ là chuỗi đã che từ, và $\theta$ là tham số của mô hình. Cơ chế này giúp mô hình học cách kết hợp cả ngữ cảnh trước và sau để dự đoán, từ đó nắm bắt biểu diễn ngữ nghĩa sâu hơn so với các mô hình một chiều.

- **Next Sentence Prediction (NSP)**: BERT còn được huấn luyện với nhiệm vụ dự đoán quan hệ giữa hai câu liên tiếp. Cho một cặp câu $(A, B)$, mô hình phải phân loại xem $B$ có phải là câu tiếp theo của $A$ trong văn bản gốc hay không. Hàm mất mát NSP được định nghĩa:

$$\mathscr{L}_{NSP} = -\big[y \log P(\text{IsNext} \mid A, B) + (1 - y) \log P(\text{NotNext} \mid A, B)\big]$$

  trong đó $y = 1$ nếu $B$ thực sự là câu tiếp theo, và $y = 0$ nếu ngược lại. Bài toán này giúp BERT học được quan hệ giữa các câu, rất quan trọng cho các tác vụ như *Question Answering* hay *Natural Language Inference*.

  Hàm mất mát tổng thể của BERT trong giai đoạn tiền huấn luyện là:

$$\mathscr{L} = \mathscr{L}_{MLM} + \mathscr{L}_{NSP}$$

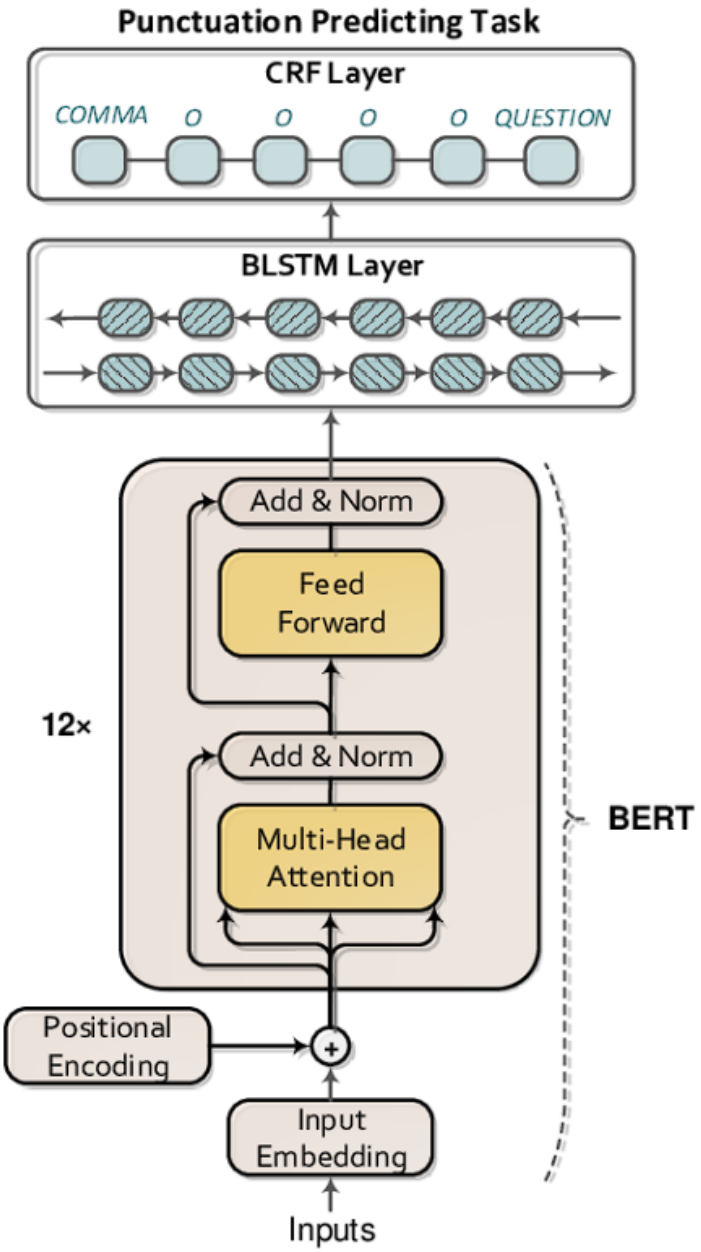
BERT được tiền huấn luyện trên một lượng dữ liệu văn bản khổng lồ (Wikipedia và BookCorpus) nhằm học các biểu diễn ngôn ngữ tổng quát. Sau đó, để áp dụng vào các tác vụ cụ thể, BERT có thể được **tinh chỉnh (fine-tuning)** bằng cách thêm một vài lớp đầu ra đặc thù cho từng bài toán. Trong quá trình fine-tuning, toàn bộ tham số của mô hình được cập nhật đồng thời, nhưng nhờ đã được tiền huấn luyện, mô hình chỉ cần ít dữ liệu hơn và huấn luyện nhanh hơn so với việc huấn luyện từ đầu. Đây là một trong những ưu điểm then chốt giúp BERT trở thành nền tảng của nhiều ứng dụng NLP hiện đại.

Kể từ khi ra đời, BERT đã đạt kết quả vượt trội (state-of-the-art) trên nhiều bộ dữ liệu chuẩn như GLUE, SQuAD và SWAG. Đồng thời, BERT đã tạo cảm hứng cho hàng loạt biến thể và cải tiến như:

- **RoBERTa** (Robustly Optimized BERT Approach): tối ưu quá trình huấn luyện, bỏ NSP và sử dụng nhiều dữ liệu hơn.
- **ALBERT** (A Lite BERT): giảm số lượng tham số bằng cách chia sẻ trọng số giữa các lớp.

- **DistilBERT**: phiên bản rút gọn của BERT giúp tăng tốc độ suy luận nhưng vẫn giữ phần lớn hiệu năng.
- **mBERT**: BERT đa ngôn ngữ, huấn luyện trên hơn 100 ngôn ngữ khác nhau.

Sự ra đời của BERT đã mở ra một kỷ nguyên mới cho NLP, thay đổi cách tiếp cận nhiều tác vụ xử lý ngôn ngữ. BERT không chỉ là một mô hình độc lập mà còn là nền tảng cho sự phát triển của các mô hình tiên tiến hơn như **T5 (Text-to-Text Transfer Transformer)**, **GPT** và các thế hệ LLM sau này, góp phần thúc đẩy mạnh mẽ ứng dụng trí tuệ nhân tạo trong thực tiễn.




Hình 1.5: Ví dụ minh họa về mô hình BERT

# CHƯƠNG 2: TỔNG QUAN MÔ HÌNH

## 2.1 Giới thiệu về SVM

Trong học máy, việc phân tích và xử lý dữ liệu là một yếu tố quan trọng để xây dựng các mô hình dự đoán. Một trong những khía cạnh phức tạp nhất của dữ liệu là đặc tính phi tuyến tính. Dữ liệu phi tuyến tính đặt ra nhiều thách thức cho các mô hình học máy tuyến tính truyền thống.

- **Dữ liệu tuyến tính**

  Dữ liệu tuyến tính là dữ liệu có thể phân tách bằng một đường thẳng (trong không gian 2D) hoặc một siêu phẳng (trong không gian nhiều chiều). Nói cách khác, các lớp dữ liệu có thể được chia tách một cách rõ ràng bằng một hàm tuyến tính.

- **Dữ liệu phi tuyến tính**

  Dữ liệu phi tuyến tính là dữ liệu mà các lớp không thể phân tách bằng một đường thẳng hoặc siêu phẳng. Trong trường hợp này, việc sử dụng các mô hình tuyến tính sẽ không hiệu quả. Các điểm dữ liệu có thể tạo thành các hình dạng phức tạp như vòng tròn, xoắn ốc, hoặc các bề mặt phi tuyến.

- **Support Vector Machine (SVM)**

  Support Vector Machine (SVM) là một thuật toán học máy thuộc loại supervised learning, được sử dụng chủ yếu cho các bài toán phân loại và hồi quy. Ý tưởng chính của SVM là tìm ra một siêu phẳng (hyperplane) tối ưu để phân tách các lớp dữ liệu. Trong không gian hai chiều, siêu phẳng này là một đường thẳng, còn trong không gian nhiều chiều, đó là một mặt phẳng hoặc một siêu phẳng.

  Trong trường hợp dữ liệu không thể phân tách tuyến tính (dữ liệu phi tuyến tính), SVM sử dụng kỹ thuật kernel để chuyển dữ liệu từ không gian gốc sang một không gian đặc trưng cao hơn, nơi dữ liệu có thể trở thành tuyến tính. Thay vì tính toán trực tiếp các tọa độ mới, SVM sử dụng một hàm kernel để tính toán sản phẩm vô hướng trong không gian đặc trưng đó.

## 2.2 SVM cho bài toán tuyến tính

Khi dữ liệu là tuyến tính, nghĩa là có thể phân tách các lớp dữ liệu bằng một đường thẳng (trong không gian 2D) hoặc một siêu phẳng (trong không gian nhiều chiều), SVM sẽ tìm cách xác định siêu phẳng này sao cho nó có margin (khoảng cách) lớn nhất giữa các lớp dữ liệu. Điều này giúp tối ưu hóa khả năng phân loại và giảm thiểu lỗi phân loại.

Giả sử chúng ta có một tập dữ liệu với hai lớp, trong đó mỗi điểm dữ liệu $\mathbf{x}_i$ thuộc về một trong hai lớp $y_i \in \{-1, 1\}$.

Mục tiêu của SVM là tìm ra siêu phẳng dưới dạng:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

Trong đó, $\mathbf{w}$ là vector trọng số và $b$ là bias. Siêu phẳng này phải đảm bảo phân tách hai lớp dữ liệu một cách chính xác, nghĩa là:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$$

Khoảng cách từ một điểm dữ liệu đến siêu phẳng được tính bằng công thức:

$$\frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

SVM sẽ tìm cách tối ưu hóa trọng số $\mathbf{w}$ và bias $b$ sao cho margin (khoảng cách giữa hai lớp) là lớn nhất. Các điểm dữ liệu gần siêu phẳng nhất được gọi là support vectors, và chính các điểm này đóng vai trò quan trọng trong việc xác định siêu phẳng phân loại.

Bài toán SVM tuyến tính được biểu diễn như một bài toán tối ưu hóa, với mục tiêu là tối ưu hóa margin giữa các lớp. Ta cần cực tiểu hóa hàm mục tiêu sau:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

với các ràng buộc:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$$

Hàm mục tiêu $\frac{1}{2}\|\mathbf{w}\|^2$ nhằm tối thiểu hóa độ lớn của vector trọng số $\mathbf{w}$, tức là tối đa hóa khoảng cách margin. Bài toán này có thể được giải bằng cách sử dụng các kỹ thuật tối ưu hóa bậc hai (Quadratic Programming).

## 2.3 Khoảng cách từ một điểm tới một siêu mặt phẳng

Trong không gian 2 chiều, khoảng cách từ một điểm có toạ độ $(x_0, y_0)$ tới đường thẳng có phương trình:

$$w_1 x + w_2 y + b = 0$$

được xác định bởi:

$$\frac{|w_1 x_0 + w_2 y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

Trong không gian 3 chiều, khoảng cách từ một điểm $(x_0, y_0, z_0)$ tới mặt phẳng:

$$w_1 x + w_2 y + w_3 z + b = 0$$

được xác định bởi:

$$\frac{|w_1 x_0 + w_2 y_0 + w_3 z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

Hơn nữa, nếu bỏ dấu trị tuyệt đối ở tử số, ta xác định được phía mà điểm đó nằm so với siêu phẳng.

- Nếu giá trị $w^T x_0 + b > 0$: điểm thuộc phía dương.
- Nếu $w^T x_0 + b < 0$: điểm thuộc phía âm.
- Nếu $w^T x_0 + b = 0$: điểm nằm trên siêu phẳng.

Tổng quát, trong không gian $d$ chiều, khoảng cách từ điểm $x_0$ tới siêu phẳng $w^T x + b = 0$ là:

$$\frac{|w^T x_0 + b|}{\|w\|_2}$$

với

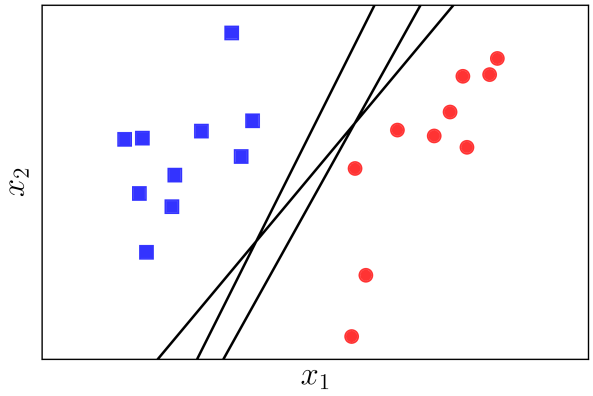
$$\|w\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$$

với d là số chiều của không gian

## 2.4 Bài toán phân chia hai lớp

Chúng ta cùng quay lại với bài toán trong *Perceptron Learning Algorithm (PLA)*. Giả sử rằng có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai classes này *linearly separable*, tức tồn tại một siêu phẳng phân chia chính xác hai classes đó. Hãy tìm một siêu mặt phẳng phân chia hai classes đó, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại.

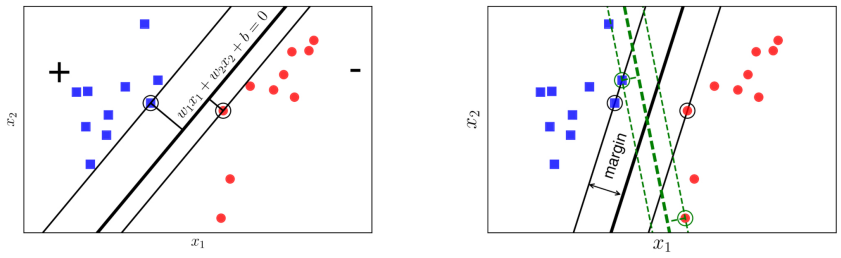
Chúng ta đã biết rằng, thuật toán PLA có thể làm được việc này nhưng nó có thể cho chúng ta vô số nghiệm như Hình 1.1 dưới đây.




Hình 1.1: Các mặt phân cách hai classes linearly separable.

Câu hỏi đặt ra là: trong vô số các mặt phân chia đó, đâu là mặt phân chia **tốt nhất** theo một tiêu chuẩn nào đó? Trong ba đường thẳng minh họa trong Hình 1.1, có hai đường thẳng khá lệch về phía class hình tròn đỏ. Điều này có thể khiến cho lớp màu đỏ không vui vì lãnh thổ xem ra bị lấn nhiều quá. Liệu có cách nào để tìm được đường phân chia mà cả hai classes đều cảm thấy công bằng và hạnh phúc nhất hay không?

Chúng ta cần tìm một tiêu chuẩn để đo sự hạnh phúc của mỗi class. Hãy xem Hình 1.2 dưới đây.




Hình 1.2: Margin của hai classes là bằng nhau và lớn nhất có thể.

Nếu ta định nghĩa mức độ hạnh phúc của một class tỉ lệ thuận với khoảng cách gần nhất từ một điểm của class đó tới đường/mặt phân chia, thì ở Hình 1.2 (trái), class tròn đỏ sẽ không được hạnh phúc cho lắm vì đường phân chia gần nó hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như thế thì mới công bằng. Khoảng cách như nhau này được gọi là **margin (lề)**.

Đã có công bằng rồi, chúng ta cần văn minh nữa. Công bằng mà cả hai đều kém hạnh phúc như nhau thì chưa phải là văn minh cho lắm.

Chúng ta xét tiếp Hình 1.2 (phải) khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, đường nào sẽ làm cho cả hai class hạnh phúc hơn? Rõ ràng đó phải là đường nét liền màu đen vì nó tạo ra một **margin rộng**

**hơn**.

Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn. Việc này, sau này các bạn sẽ thấy, là một điểm khá quan trọng giúp *Support Vector Machine (SVM)* mang lại kết quả phân loại tốt hơn so với *Neural Network với 1 layer*, tức *Perceptron Learning Algorithm*.

Bài toán tối ưu trong *Support Vector Machine (SVM)* chính là bài toán đi tìm đường phân chia sao cho **margin là lớn nhất**. Đây cũng là lý do vì sao SVM còn được gọi là **Maximum Margin Classifier**. Nguồn gốc của tên gọi *Support Vector Machine* sẽ sớm được làm sáng tỏ.

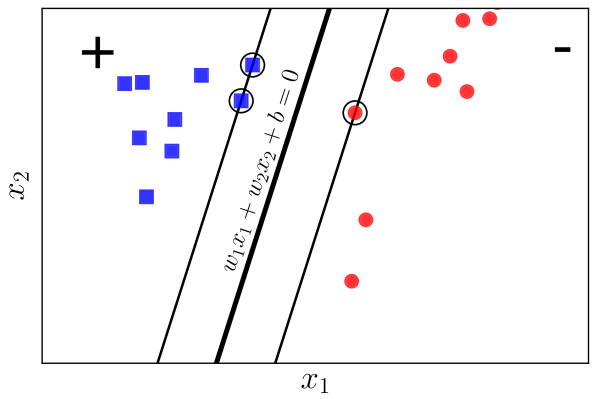
## 2.5 Xây dựng bài toán tối ưu cho SVM

Giả sử rằng các cặp dữ liệu của training set là

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

với vector $x_i \in \mathbb{R}^d$ thể hiện đầu vào của một điểm dữ liệu và $y_i$ là nhãn của điểm dữ liệu đó. $d$ là số chiều của dữ liệu và $N$ là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2) giống như trong PLA.

Để giúp các bạn dễ hình dung, chúng ta cùng xét trường hợp trong không gian hai chiều dưới đây. Không gian hai chiều để các bạn dễ hình dung, các phép toán hoàn toàn có thể được tổng quát lên không gian nhiều chiều.




Hình 1.3: Phân tích bài toán SVM.

Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt

$$w^T x + b = w_1 x_1 + w_2 x_2 + b = 0$$

là mặt phân chia giữa hai classes. Hơn nữa, class 1 nằm về phía dương, class -1 nằm về phía âm của mặt phân

chia. Nếu ngược lại, ta chỉ cần đổi dấu của $w$ và $b$. Chú ý rằng chúng ta cần đi tìm các hệ số $w$ và $b$.

Ta quan sát thấy một điểm quan trọng sau đây: với cặp dữ liệu $(x_n, y_n)$ bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(w^T x_n + b)}{\|w\|^2}$$

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, $y_n$ luôn cùng dấu với phía của $x_n$. Từ đó suy ra $y_n$ cùng dấu với $(w^T x_n + b)$, và tử số luôn là một số không âm.

Với mặt phân chia như trên, margin được tính là khoảng cách gần nhất từ một điểm tới mặt đó (bất kể điểm nào trong hai classes):

$$\text{margin} = \min_n \frac{y_n(w^T x_n + b)}{\|w\|^2}$$

Bài toán tối ưu trong SVM chính là bài toán tìm $w$ và $b$ sao cho margin này đạt giá trị lớn nhất:

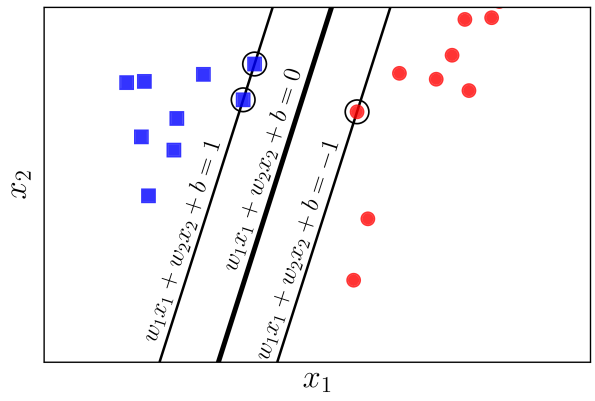
$$(w, b) = \arg\max_{w,b} \left\{ \min_n \frac{y_n(w^T x_n + b)}{\|w\|^2} \right\} = \arg\max_{w,b} \left\{ \frac{1}{\|w\|^2} \min_n \left( y_n(w^T x_n + b) \right) \right\} \tag{1}$$

Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng có cách để đưa nó về bài toán đơn giản hơn.

Nhận xét quan trọng nhất là nếu ta thay vector hệ số $w$ bởi $kw$ và $b$ bởi $kb$ trong đó $k$ là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi. Dựa trên tính chất này, ta có thể giả sử:

$$y_n(w^T x_n + b) = 1$$

với những điểm nằm gần mặt phân chia nhất như Hình 1.4 dưới đây.




Hình 1.4: Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn.

Như vậy, với mọi $n$, ta có:

$$y_n(w^T x_n + b) \geq 1$$

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

$$(w,b) = \arg\max_{w,b} \frac{1}{\|w\|^2}$$

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

Bằng một biến đổi đơn giản, ta có thể đưa bài toán này về:

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

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

Ở đây, chúng ta đã lấy nghịch đảo hàm mục tiêu, bình phương nó để được một hàm khả vi, và nhân với $\frac{1}{2}$ để biểu thức đạo hàm đẹp hơn.

Quan sát quan trọng: Trong bài toán (3), hàm mục tiêu là một norm, nên là một hàm lồi. Các hàm bất đẳng thức ràng buộc là các hàm tuyến tính theo $w$ và $b$, nên chúng cũng là các hàm lồi. Vậy bài toán tối ưu (3) có hàm mục tiêu là lồi, và các hàm ràng buộc cũng là lồi, nên nó là một bài toán lồi. Hơn nữa, nó là một Quadratic Programming. Thậm chí, hàm mục tiêu là strictly convex vì $\|w\|^2 = w^T I w$ và $I$ là ma trận đơn vị - một ma trận xác định dương. Từ đây có thể suy ra nghiệm cho SVM là duy nhất.

Đến đây thì bài toán này có thể giải được bằng các công cụ hỗ trợ tìm nghiệm cho Quadratic Programing, ví dụ CVXOPT.

Tuy nhiên, việc giải bài toán này trở nên phức tạp khi số chiều $d$ của không gian dữ liệu và số điểm dữ liệu $N$ tăng lên cao.

Người ta thường giải bài toán đối ngẫu của bài toán này. Thứ nhất, bài toán đối ngẫu có những tính chất thú vị hơn khiến nó được giải hiệu quả hơn. Thứ hai, trong quá trình xây dựng bài toán đối ngẫu, người ta thấy rằng SVM có thể được áp dụng cho những bài toán mà dữ liệu không linearly separable, tức các đường phân chia không phải là một mặt phẳng mà có thể là các mặt có hình thù phức tạp hơn.

Đến đây, bạn đọc có thể bắt đầu hiểu tại sao tôi cần viết 3 bài 16–18 trước khi viết bài này. Nếu bạn muốn hiểu sâu hơn về SVM, tôi khuyến khích đọc Mục 3 dưới đây. Nếu không, bạn có thể sang Mục 4 để xem ví dụ về cách sử dụng SVM khi lập trình.

## Xác định class cho một điểm dữ liệu mới

Sau khi tìm được mặt phân cách $w^T x + b = 0$, class của bất kỳ một điểm nào sẽ được xác định đơn giản bằng cách:

$$\text{class}(x) = \text{sgn}(w^T x + b)$$

Trong đó hàm sgn là hàm xác định dấu, nhận giá trị 1 nếu đối số là không âm và $-1$ nếu ngược lại.

## 2.6 Bài toán đối ngẫu cho SVM

Nếu đẳng thức $p^* = d^*$ thoả mãn, the optimal duality gap bằng không, ta nói rằng strong duality xảy ra. Lúc này, việc giải bài toán đối ngẫu đã giúp ta tìm được chính xác giá trị tối ưu của bài toán gốc.

Thật không may, strong duality không thường xuyên xảy ra trong các bài toán tối ưu. Tuy nhiên, nếu bài toán gốc là lồi, tức có dạng:

$$x = \arg\min_x f_0(x) \quad \text{subject to: } f_i(x) \leq 0, i = 1,2,\ldots,m \quad \text{và } Ax = b \tag{2.1}$$

trong đó $f_0, f_1, \ldots, f_m$ là các hàm lồi, chúng ta thường (không luôn luôn) có strong duality. Có rất nhiều nghiên cứu thiết lập các điều kiện, ngoài tính chất lồi, để strong duality xảy ra. Những điều kiện đó thường có tên là constraint qualifications.

Một trong các constraint qualification đơn giản nhất là Slater's condition.

**Định nghĩa:** Một điểm feasible của bài toán (12) được gọi là strictly feasible nếu:

$$f_i(x) < 0, \quad i = 1,2,\ldots,m, \quad Ax = b$$

**Định lý Slater:** Nếu tồn tại một điểm strictly feasible (và bài toán gốc là lồi), thì strong duality xảy ra.

Điều kiện khá đơn giản sẽ giúp ích cho nhiều bài toán tối ưu sau này. Với các bài toán lồi và strong duality xảy ra, các điều kiện KKT phía trên cũng là điều kiện đủ. Vậy với các bài toán lồi với hàm mục tiêu và hàm ràng buộc là khả vi, bất kỳ điểm nào thoả mãn các điều kiện KKT đều là primal và dual optimal của bài toán gốc và bài toán đối ngẫu.

Từ đây ta có thể thấy rằng: Với một bài toán lồi và điều kiện Slater thoả mãn (suy ra strong duality) thì các điều kiện KKT là điều cần và đủ của nghiệm.

Các điều kiện KKT rất quan trọng trong tối ưu. Trong một vài trường hợp đặc biệt (chúng ta sẽ thấy trong bài Support Vector Machine sắp tới), việc giải hệ (bất) phương trình các điều kiện KKT là khả thi. Rất nhiều các thuật toán tối ưu được xây dựng giả trên việc giải hệ điều kiện KKT.

**Ví dụ: Equality constrained convex quadratic minimization.**

Xét bài toán:

$$x = \arg\min_x \frac{1}{2}x^T Px + q^T x + r \quad \text{subject to: } Ax = b$$

trong đó $P \in S_+^n$ (tập các ma trận đối xứng nửa xác định dương).

Lagrangian:

$$L(x, \nu) = \frac{1}{2}x^T Px + q^T x + r + \nu^T (Ax - b)$$

Điều kiện KKT cho bài toán này là:

$$Ax^* = b$$

$$Px^* + q + A^T \nu^* = 0$$

Phương trình thứ hai chính là phương trình đạo hàm của Lagrangian tại $x^*$ bằng 0.

Hệ phương trình này có thể được viết lại đơn giản là:

$$\begin{bmatrix} P & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}$$

Đây là một phương trình tuyến tính đơn giản!

Nhắc lại rằng bài toán tối ưu (3) là một bài toán lồi. Chúng ta biết rằng: nếu một bài toán lồi thoả mãn tiêu chuẩn Slater thì strong duality thoả mãn. Và nếu strong duality thoả mãn thì nghiệm của bài toán chính là nghiệm của hệ điều kiện KKT.

—

### 2.6.1 Kiểm tra tiêu chuẩn Slater

Bước tiếp theo, chúng ta sẽ chứng minh bài toán tối ưu (3) thoả mãn điều kiện Slater. Điều kiện Slater nói rằng, nếu tồn tại $w, b$ thoả mãn:

$$1 - y_n(w^T x_n + b) < 0, \quad \forall n = 1, 2, \ldots, N$$

thì strong duality thoả mãn.

Việc kiểm tra này tương đối đơn giản. Vì ta biết rằng luôn luôn có một (siêu) mặt phẳng phân chia hai classes nếu hai class đó là linearly separable, tức bài toán có nghiệm, nên feasible set của bài toán tối ưu (3) phải khác rỗng.

Tức luôn luôn tồn tại cặp $(w_0, b_0)$ sao cho:

$$1 - y_n(w_0^T x_n + b_0) \leq 0, \quad \forall n = 1, 2, \ldots, N$$

$$\Leftrightarrow 2 - y_n(2w_0^T x_n + 2b_0) \leq 0, \quad \forall n = 1, 2, \ldots, N$$

Vậy chỉ cần chọn $w_1 = 2w_0$ và $b_1 = 2b_0$, ta sẽ có:

$$1 - y_n(w_1^T x_n + b_1) \leq -1 < 0, \quad \forall n = 1, 2, \ldots, N$$

Từ đó suy ra điều kiện Slater thoả mãn.

—

### 2.6.2 Lagrangian của bài toán SVM

Xét một bài toán tối ưu tổng quát dưới dạng:

$$x^* = \arg\min_x f_0(x) \quad \text{subject to: } f_i(x) \leq 0,\ i = 1,2,\ldots,m, \quad g_j(x) = 0,\ j = 1,2,\ldots,p \tag{9}$$

với miền xác định:

$$D = \left(\bigcap_{i=0}^{m} \operatorname{dom} f_i\right) \cap \left(\bigcap_{j=1}^{p} \operatorname{dom} g_j\right).$$

Xin lưu ý rằng, chúng ta không giả sử tính chất lồi của hàm mục tiêu hay các hàm ràng buộc ở đây. Giả sử duy nhất là $D \neq \varnothing$ (tập xác định không rỗng).

Lagrangian tổng quát được xây dựng bằng cách kết hợp các nhân tử Lagrange cho từng ràng buộc:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j g_j(x), \tag{2.2}$$

với $\lambda = [\lambda_1, \ldots, \lambda_m]^T$, $\nu = [\nu_1, \ldots, \nu_p]^T$ (ký hiệu $\nu$ là chữ cái Hy Lạp "nu"). Các vector này được gọi là *biến đối ngẫu* (dual variables) hoặc *nhân tử Lagrange*. Nếu $x \in \mathbb{R}^n$, thì tổng số biến của Lagrangian là $n+m+p$.

—

**Lagrangian của bài toán SVM** Áp dụng cho bài toán tối ưu (3), ta có:

$$L(w, b, \lambda) = \frac{1}{2}\|w\|_2^2 + \sum_{n=1}^{N} \lambda_n \big(1 - y_n(w^T x_n + b)\big) \tag{4}$$

với $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T$ và $\lambda_n \geq 0$, $\forall n = 1,2,\ldots,N$.

—

**Hàm đối ngẫu (Dual Function)** Hàm đối ngẫu được định nghĩa là:

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left( f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j g_j(x) \right).$$

Nếu $L(x, \lambda, \nu)$ không bị chặn dưới theo $x$, thì $g(\lambda, \nu) = -\infty$.

—

**Một số điểm quan trọng:**

- Lấy inf trên miền $x \in D$ (tập xác định của bài toán). Lưu ý rằng tập khả thi thường là tập con của miền xác định $D$.
- Với mỗi $x$, $L(x, \lambda, \nu)$ là một hàm affine của $(\lambda, \nu)$, tức là một hàm lõm. Do đó, hàm đối ngẫu $g(\lambda, \nu)$ chính là pointwise infimum của (có thể vô hạn) các hàm affine, nên nó luôn là một *hàm lõm*.
- Điều này đúng ngay cả khi bài toán gốc không phải là bài toán lồi.

Hàm đối ngẫu là công cụ quan trọng để xây dựng bài toán đối ngẫu trong SVM.

### 2.6.3 Hàm đối ngẫu Lagrange

Hàm đối ngẫu Lagrange được định nghĩa là:

$$g(\lambda) = \min_{w,b} L(w, b, \lambda) \quad \text{với } \lambda \succeq 0$$

Việc tìm giá trị nhỏ nhất của hàm này theo $w$ và $b$ có thể được thực hiện bằng cách giải hệ phương trình đạo hàm của $L(w, b, \lambda)$ theo $w$ và $b$ bằng 0:

$$\frac{\partial L(w, b, \lambda)}{\partial w} = w - \sum_{n=1}^{N} \lambda_n y_n x_n = 0 \quad \Rightarrow \quad w = \sum_{n=1}^{N} \lambda_n y_n x_n \tag{5}$$

$$\frac{\partial L(w, b, \lambda)}{\partial b} = -\sum_{n=1}^{N} \lambda_n y_n = 0 \tag{6}$$

Thay (5) và (6) vào (4) ta thu được $g(\lambda)$ (phần này tôi rút gọn, coi như một bài tập nhỏ):

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m x_n^T x_m \tag{7}$$

Xét ma trận:

$$V = [y_1 x_1, y_2 x_2, \ldots, y_N x_N]$$

và vector $1 = [1, 1, \ldots, 1]^T$, ta có thể viết lại:

$$g(\lambda) = -\frac{1}{2} \lambda^T V^T V \lambda + 1^T \lambda \tag{8}$$

Đặt $K = V^T V$, ta có một quan sát: $K$ là một ma trận nửa xác định dương.

### 2.6.4 Bài toán đối ngẫu Lagrange

Từ đó, kết hợp hàm đối ngẫu Lagrange và các điều kiện ràng buộc của $\lambda$, ta thu được bài toán đối ngẫu Lagrange:

$$\lambda^* = \arg\max_{\lambda} g(\lambda) \quad \text{subject to: } \lambda \succeq 0, \quad \sum_{n=1}^{N} \lambda_n y_n = 0 \tag{9}$$

Ràng buộc thứ hai được lấy từ (6).

Đây là một bài toán lồi vì ta đang đi tìm giá trị lớn nhất của một hàm mục tiêu concave trên một polyhedron. Bài toán này cũng được gọi là một *Quadratic Programming* (QP) và có thể giải được bằng các thư viện như `CVXOPT`.

Trong bài toán đối ngẫu này, số tham số (parameters) cần tìm là $N$, chính là chiều của $\lambda$, tức số điểm dữ liệu. Trong khi đó, với bài toán gốc (3), số tham số cần tìm là $d+1$, gồm $w \in \mathbb{R}^d$ và $b \in \mathbb{R}$. Trong rất nhiều trường hợp, số điểm dữ liệu trong tập huấn luyện lớn hơn số chiều dữ liệu rất nhiều. Nếu giải trực tiếp bằng các công cụ giải Quadratic Programming, có thể bài toán đối ngẫu còn phức tạp (tốn thời gian) hơn so với bài toán gốc.

Tuy nhiên, điều hấp dẫn của bài toán đối ngẫu này đến từ phần *Kernel Support Vector Machine* (Kernel SVM), áp dụng cho các bài toán mà dữ liệu không linearly separable hoặc gần linearly separable. Phần Kernel SVM sẽ được trình bày ở các phần tiếp theo. Ngoài ra, dựa vào tính chất đặc biệt của hệ điều kiện KKT, SVM có thể được giải bằng nhiều phương pháp hiệu quả hơn.

### 2.6.5 Điều kiện KKT

Quay trở lại bài toán, vì đây là một bài toán lồi và strong duality thoả mãn, nghiệm của bài toán sẽ thoả mãn hệ điều kiện KKT sau đây với biến số là $w$, $b$ và $\lambda$:

$$1 - y_n(w^T x_n + b) \leq 0, \qquad \forall n = 1,2,\ldots,N \tag{10}$$

$$\lambda_n \geq 0, \qquad \forall n = 1,2,\ldots,N \tag{2.3}$$

$$\lambda_n(1 - y_n(w^T x_n + b)) = 0, \qquad \forall n = 1,2,\ldots,N \tag{11}$$

$$w = \sum_{n=1}^{N} \lambda_n y_n x_n \tag{12}$$

$$\sum_{n=1}^{N} \lambda_n y_n = 0 \tag{13}$$

Trong những điều kiện trên, điều kiện (11) là thú vị nhất. Từ đó ta có thể suy ra ngay, với $n$ bất kỳ, hoặc $\lambda_n = 0$ hoặc $1 - y_n(w^T x_n + b) = 0$. Trường hợp thứ hai tương đương với:

$$w^T x_n + b = y_n, \quad \text{với } y_n^2 = 1, \forall n \tag{14}$$

Những điểm thoả mãn (14) chính là những điểm nằm gần mặt phân chia nhất, là những điểm được khoanh tròn trong Hình 4. Hai đường thẳng $w^T x_n + b = \pm 1$ tựa lên các điểm này. Những điểm này còn được gọi là *Support Vectors*, từ đó cái tên Support Vector Machine ra đời.

Một quan sát khác, số lượng điểm thoả mãn (14) thường rất ít so với $N$. Hầu hết các $\lambda_n$ bằng 0, nên vector $\lambda \in \mathbb{R}^N$ là một sparse vector. Do đó, SVM được xếp vào *Sparse Models*, vốn có cách giải hiệu quả hơn các mô hình dense.

Với tập dữ liệu nhỏ, có thể giải hệ KKT bằng cách xét $\lambda_n = 0$ hoặc $\lambda_n \neq 0$, nhưng tổng số trường hợp là $2^N$, rất lớn nếu $N > 50$, do đó không khả thi. Trong phần tiếp theo, chúng ta sẽ giải bài toán tối ưu (9) bằng `CVXOPT` hoặc `sklearn`.

Sau khi tìm được $\lambda$ từ bài toán (9), ta suy ra:

$$w = \sum_{m \in S} \lambda_m y_m x_m \tag{16}$$

$$b = \frac{1}{N_S} \sum_{n \in S} \left( y_n - w^T x_n \right) = \frac{1}{N_S} \sum_{n \in S} \left( y_n - \sum_{m \in S} \lambda_m y_m x_m^T x_n \right) \tag{15}$$

với $S = \{n : \lambda_n \neq 0\}$ và $N_S$ là số phần tử của $S$.

Cuối cùng, để xác định class của một điểm $x$ mới, ta dùng biểu thức:

$$\text{class}(x) = \text{sgn}\left( w^T x + b \right) = \text{sgn}\left( \sum_{m \in S} \lambda_m y_m x_m^T x + \frac{1}{N_S} \sum_{n \in S} \left( y_n - \sum_{m \in S} \lambda_m y_m x_m^T x_n \right) \right) \tag{2.4}$$

Biểu thức này phụ thuộc vào tích vô hướng giữa $x$ và từng $x_n \in S$, và sẽ trở nên quan trọng trong Kernel SVM.

### 2.6.6 SVM với Soft Margin

Trong thực tế, dữ liệu có thể không hoàn toàn tuyến tính và đôi khi có nhiễu, dẫn đến việc không thể phân tách chính xác các lớp dữ liệu bằng một siêu phẳng. Để giải quyết vấn đề này, SVM tuyến tính sử dụng một biến số slack $\xi_i$ để cho phép một số điểm dữ liệu nằm trong margin hoặc bị phân loại sai. Mô hình này được gọi là Soft Margin SVM.

Bài toán tối ưu hóa của Soft Margin SVM được viết lại như sau:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

với các ràng buộc:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i \quad \text{và} \quad \xi_i \geq 0$$

Trong đó, $C$ là một tham số điều chỉnh, giúp cân bằng giữa việc tối thiểu hóa lỗi phân loại và tối đa

hóa margin. Khi $C$ lớn, mô hình sẽ cố gắng giảm thiểu lỗi phân loại, nhưng có thể dẫn đến overfitting. Ngược lại, khi $C$ nhỏ, mô hình sẽ tập trung vào việc tối đa hóa margin nhưng có thể chấp nhận một số lỗi phân loại.

## 2.7 Ưu điểm và nhược điểm

### 2.7.1 Ưu điểm của SVM

- **Hiệu quả cao với dữ liệu có số lượng đặc trưng lớn**: SVM hoạt động tốt khi số lượng đặc trưng (features) của dữ liệu lớn hơn nhiều so với số lượng mẫu dữ liệu.
- **Hiệu suất cao với dữ liệu phi tuyến tính**: Khi dữ liệu không thể phân tách tuyến tính, SVM có thể sử dụng các kỹ thuật kernel để chuyển đổi dữ liệu sang không gian có thể phân tách.
- **Tối ưu hóa margin**: SVM tối đa hóa margin giữa các lớp dữ liệu, giúp cải thiện khả năng tổng quát hóa (generalization) của mô hình.
- **Lời giải duy nhất và tối ưu**: Bài toán tối ưu hóa của SVM có lời giải duy nhất và tối ưu nhờ vào việc sử dụng kỹ thuật tối ưu hóa bậc hai (Quadratic Programming).
- **Xử lý hiệu quả với outliers**: SVM có khả năng xử lý tốt các dữ liệu ngoại lai (outliers) bằng cách sử dụng Soft Margin.

### 2.7.2 Nhược điểm của SVM

- **Khó khăn trong việc chọn kernel phù hợp**: Một trong những thách thức lớn nhất khi sử dụng SVM là lựa chọn kernel phù hợp cho dữ liệu.
- **Độ phức tạp tính toán cao**: SVM thường yêu cầu tính toán phức tạp, đặc biệt là với các bộ dữ liệu lớn.
- **Nhạy cảm với dữ liệu nhiễu**: Mặc dù SVM có khả năng xử lý outliers tốt, nhưng nó vẫn có thể bị ảnh hưởng bởi các dữ liệu nhiễu.
- **Thời gian huấn luyện lâu**: Với các bộ dữ liệu lớn hoặc có nhiều đặc trưng, thời gian huấn luyện của SVM có thể rất lâu.
- **Khó mở rộng với nhiều lớp**: Việc mở rộng SVM cho bài toán phân loại đa lớp (multi-class classification) có thể trở nên phức tạp và kém hiệu quả hơn so với các thuật toán khác.

## 2.8 Ứng dụng trong NPL

Phân loại tin tức là bài toán gán nhãn cho các bài báo hoặc tin tức theo các chủ đề đã được định sẵn, ví dụ như chính trị, thể thao, kinh tế hay giải trí. Mỗi bài báo được biểu diễn bằng một vector đặc trưng số, trong

đó các thành phần của vector thể hiện các thông tin về nội dung văn bản, có thể là số lần xuất hiện của từ, trọng số TF-IDF hoặc embeddings từ các mô hình học sâu như BERT. Nhãn của mỗi bài báo xác định lớp mà bài báo thuộc về, ví dụ chính trị hoặc thể thao. Tập dữ liệu huấn luyện bao gồm nhiều cặp dữ liệu, mỗi cặp gồm một vector đặc trưng và nhãn tương ứng.

Trước khi áp dụng SVM, dữ liệu văn bản cần được chuyển đổi thành dạng vector số. Các phương pháp phổ biến gồm Bag-of-Words, TF-IDF và các embeddings ngữ nghĩa. Bag-of-Words đếm số lần xuất hiện của mỗi từ trong bài báo và sử dụng thông tin này để tạo vector. Phương pháp TF-IDF cải tiến bằng cách gán trọng số cho các từ dựa trên tần suất xuất hiện trong bài báo so với toàn bộ tập dữ liệu, giúp tăng khả năng phân biệt các từ quan trọng. Các embeddings như BERT hoặc Word2Vec biến mỗi từ hoặc câu thành vector ngữ nghĩa trong không gian đa chiều, nhờ đó mô hình có thể nắm bắt được mối quan hệ ngữ cảnh giữa các từ. Sau bước biểu diễn này, mỗi bài báo được đại diện bởi một vector số trong không gian nhiều chiều, sẵn sàng để đưa vào mô hình SVM.

Mô hình SVM được sử dụng nhằm tìm một mặt phân cách tối ưu giữa các lớp, sao cho khoảng cách giữa các điểm dữ liệu gần nhất và mặt phân chia được tối đa hóa. Với bài toán phân loại tin tức nhiều lớp, SVM nhị phân có thể được mở rộng bằng các chiến lược one-vs-rest hoặc one-vs-one. Hàm mục tiêu của SVM bao gồm việc tối thiểu hóa chuẩn của vector trọng số đồng thời phạt các lỗi phân loại, được điều chỉnh bởi một siêu tham số $C$. Trong trường hợp dữ liệu không phân tách tuyến tính hoàn toàn, các kernel như RBF hoặc polynomial có thể được áp dụng, biến không gian đặc trưng để SVM có thể phân tách các lớp một cách phi tuyến.

Quá trình huấn luyện SVM bắt đầu bằng việc chia dữ liệu thành tập huấn luyện và tập kiểm tra. Tất cả các bài báo đều được biểu diễn thành vector số, sau đó mô hình SVM được huấn luyện trên tập huấn luyện. Việc lựa chọn siêu tham số và kernel thường được thực hiện thông qua cross-validation để tối ưu hiệu quả phân loại. Sau khi mô hình huấn luyện xong, hiệu quả được đánh giá trên tập kiểm tra thông qua các chỉ số như accuracy, precision, recall và F1-score.

Khi dự đoán một bài báo mới, SVM tính toán điểm số dựa trên vector đặc trưng của bài báo và xác định nhãn dựa trên dấu của biểu thức tuyến tính trong trường hợp nhị phân hoặc chọn lớp có giá trị score cao nhất trong chiến lược one-vs-rest. Một trong những ưu điểm lớn của SVM trong phân loại tin tức là khả năng làm việc hiệu quả với dữ liệu nhiều chiều như TF-IDF, hỗ trợ các kernel để xử lý quan hệ phi tuyến giữa các từ và chỉ dựa vào các support vectors quan trọng để xác định mặt phân chia, giúp mô hình trở nên sparse và tiết kiệm bộ nhớ. Nhược điểm là việc huấn luyện có thể chậm nếu số lượng dữ liệu lớn và khó giải thích trực quan đối với các bài toán nhiều lớp.

Nhờ những đặc điểm này, SVM vẫn là một trong những mô hình hiệu quả cho các bài toán phân loại văn bản, đặc biệt khi số lượng từ vựng lớn và mối quan hệ giữa các từ cần được mô hình hóa trong không gian đa chiều.

# CHƯƠNG 3: ỨNG DỤNG MÔ HÌNH SVM TRONG BÀI TOÁN PHÂN LOẠI TIN

## 3.1 Mô tả bài toán

Bài toán phân loại tin tức là một bài toán trong lĩnh vực xử lý ngôn ngữ tự nhiên, với mục tiêu dự đoán thể loại của một bài báo dựa trên tiêu đề và nội dung của nó. Trong nghiên cứu này, chúng tôi sử dụng bộ dữ liệu AG News, một bộ dữ liệu phổ biến bao gồm các bài báo thuộc bốn lớp: Thế giới, Thể thao, Kinh doanh và Khoa học/Công nghệ. Mỗi bài báo được gán nhãn tương ứng, với tổng số bài báo trong bộ dữ liệu đủ lớn để huấn luyện và đánh giá các mô hình học máy. Cho trước tập huấn luyện gồm các cặp dữ liệu $(x_i, y_i)$, trong đó $x_i$ là văn bản của bài báo và $y_i$ là nhãn lớp, nhiệm vụ là xây dựng một mô hình học máy $f$ sao cho với bất kỳ bài báo mới $x_{\text{new}}$, mô hình có thể dự đoán chính xác nhãn $\hat{y}_{\text{new}} = f(x_{\text{new}})$.

Trong nghiên cứu này, chúng tôi áp dụng mô hình Support Vector Machine (SVM) để thực hiện phân loại. Quá trình xây dựng mô hình SVM bắt đầu bằng việc tiền xử lý dữ liệu văn bản, bao gồm loại bỏ các ký tự đặc biệt, chuyển tất cả các chữ về dạng thường, loại bỏ các từ dừng, đồng thời chuẩn hóa dữ liệu nhằm đảm bảo tính đồng nhất. Sau đó, văn bản được biểu diễn dưới dạng vector TF-IDF, mô hình hóa tần suất xuất hiện của các từ có chuẩn hóa, giúp dữ liệu văn bản có thể sử dụng trực tiếp trong SVM. Mô hình SVM với kernel tuyến tính được huấn luyện để tìm siêu phẳng phân loại tối ưu giữa các lớp, tối đa hóa khoảng cách giữa các điểm dữ liệu thuộc các lớp khác nhau và từ đó nâng cao khả năng phân loại chính xác. Hiệu suất của mô hình được đánh giá thông qua các chỉ số như độ chính xác (accuracy), độ chính xác trung bình (precision), độ nhạy (recall), F1-score, cùng với ma trận nhầm lẫn (confusion matrix) để phân tích chi tiết khả năng phân loại từng lớp.

Việc sử dụng SVM cho bài toán phân loại tin tức dựa trên TF-IDF có ưu điểm là mô hình đơn giản, dễ huấn luyện và hiệu quả với dữ liệu dạng thưa, đồng thời xác định các support vectors, tức những bài báo quan trọng nhất quyết định siêu phẳng phân lớp, giúp mô hình gọn nhẹ và dễ giải thích. Bài toán này có ứng dụng thực tiễn cao trong việc tự động phân loại bài báo theo chủ đề, hỗ trợ các hệ thống tin tức, nâng cao trải nghiệm người đọc và là nền tảng để phát triển các hệ thống gợi ý thông minh.

## 3.2 Mô tả dữ liệu và Tiền xử lý dữ liệu

### 3.2.1 Mô tả dữ liệu

Bộ dữ liệu **AG News** là một tập dữ liệu phổ biến trong lĩnh vực phân loại văn bản và xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Nó được xây dựng từ AG's corpus of news articles, chứa các bài báo được phân loại vào bốn chủ đề chính, bao gồm *World* (Thế giới), *Sports* (Thể thao), *Business* (Kinh doanh), và *Sci/Tech* (Khoa học và Công nghệ).

Tập huấn luyện bao gồm 120.000 bài báo, mỗi lớp có 30.000 bài, trong khi tập kiểm tra có 7.600 bài báo, với mỗi lớp 1.900 bài. Mỗi bài báo trong bộ dữ liệu có ba trường thông tin chính:

- `class`: nhãn phân loại của bài báo, có giá trị từ 1 đến 4, tương ứng với các lớp World, Sports, Business, Sci/Tech.
- `title`: tiêu đề của bài báo.
- `description`: mô tả ngắn gọn về nội dung bài báo.

Nhìn chung, tập dữ liệu này cung cấp một lượng lớn thông tin văn bản ngắn, phù hợp cho các mô hình học máy phân loại văn bản. Dữ liệu được cung cấp dưới định dạng CSV, với mỗi dòng đại diện cho một bài báo.

Một số thống kê sơ bộ về dữ liệu cho thấy độ dài trung bình của tiêu đề khoảng 11 từ, còn mô tả có độ dài trung bình khoảng 35 từ. Sự phân bố số lượng bài báo giữa các lớp là đồng đều, giúp tránh tình trạng lệch lớp (class imbalance).

### 3.2.2 Phân tích trực quan dữ liệu

Để hiểu rõ hơn về tập dữ liệu, chúng ta có thể trực quan hóa số lượng bài báo trong mỗi lớp. Biểu đồ cột hoặc histogram sẽ cho thấy rằng mỗi lớp đều chiếm khoảng 25% tổng số bài báo, xác nhận rằng dữ liệu cân bằng giữa các lớp.

Ngoài ra, có thể sử dụng *Word Cloud* để quan sát các từ phổ biến trong từng lớp. Ví dụ, lớp *Sports* thường xuất hiện các từ như “game”, “team”, “player”, trong khi lớp *Business* xuất hiện các từ “market”, “company”, “stock”. Những trực quan này giúp chúng ta hình dung rõ ràng các đặc trưng văn bản phân biệt các lớp.

### 3.2.3 Tiền xử lý dữ liệu

Để sử dụng dữ liệu văn bản với mô hình học máy, đặc biệt là SVM, cần thực hiện các bước tiền xử lý dữ liệu nhằm chuẩn hóa và loại bỏ nhiễu. Các bước chi tiết được thực hiện như sau:

**Kết hợp tiêu đề và mô tả:** Tiêu đề và mô tả được kết hợp thành một chuỗi văn bản duy nhất cho mỗi bài báo để mô hình có thể học được nhiều thông tin hơn từ toàn bộ bài báo, không chỉ riêng tiêu đề hoặc mô tả.

**Chuyển đổi chữ thường:** Tất cả văn bản được chuyển sang dạng chữ thường để loại bỏ sự khác biệt giữa chữ hoa và chữ thường, tránh tình trạng cùng một từ nhưng bị nhận diện khác nhau do kiểu chữ.

**Loại bỏ dấu câu và ký tự đặc biệt:** Các ký tự như dấu chấm, dấu phẩy, dấu hỏi hay các ký tự không thuộc bảng chữ cái được loại bỏ. Điều này giúp giảm nhiễu và tập trung vào từ ngữ mang thông tin phân loại.

**Loại bỏ từ dừng (stopwords):** Các từ phổ biến nhưng không mang nhiều thông tin phân loại, ví dụ như “the”, “is”, “of”, được loại bỏ. Việc này giúp giảm kích thước dữ liệu và tăng tính hiệu quả của các đặc trưng học được từ văn bản.

**Chuẩn hóa từ (lemmatization):** Từ được chuẩn hóa về dạng gốc (lemma). Ví dụ, các từ "running", "ran" đều được chuẩn hóa về "run". Việc này giúp giảm thiểu sự đa dạng về dạng từ, tăng tính thống nhất của dữ liệu, và cải thiện khả năng học của mô hình.

**Mã hóa nhãn:** Nhãn văn bản (class) được chuyển thành dạng số nguyên từ 0 đến 3 để phù hợp với các mô hình học máy.

### 3.2.4 Trích xuất đặc trưng

Sau khi tiền xử lý văn bản, bước tiếp theo là trích xuất đặc trưng. Trong nghiên cứu này, chúng tôi sử dụng phương pháp **TF-IDF (Term Frequency - Inverse Document Frequency)** để biến đổi văn bản thành dạng vector số:

- TF-IDF giúp đánh giá tầm quan trọng của một từ trong một tài liệu dựa trên tần suất xuất hiện trong tài liệu đó và trong toàn bộ tập dữ liệu.
- Mỗi bài báo được biểu diễn bằng một vector TF-IDF với chiều dài bằng số từ xuất hiện trong tập dữ liệu (ở đây giới hạn tối đa là 5000 từ phổ biến nhất).

Điều này cho phép mô hình SVM học được các đặc trưng quan trọng từ văn bản và phân loại bài báo dựa trên sự xuất hiện của các từ khóa.

Tập dữ liệu sau khi tiền xử lý và trích xuất đặc trưng được chia thành tập huấn luyện và tập kiểm tra theo tỷ lệ 80/20. Việc chia này được thực hiện theo phương pháp *stratified split* để đảm bảo tỷ lệ các lớp được giữ nguyên trong cả hai tập, từ đó đánh giá chính xác hiệu suất mô hình.

Như vậy, sau khi hoàn tất các bước tiền xử lý và trích xuất đặc trưng, dữ liệu đã sẵn sàng để huấn luyện mô hình SVM. Các bước này giúp loại bỏ nhiễu, chuẩn hóa văn bản và tạo ra các đặc trưng số đại diện cho thông tin ngữ nghĩa trong bài báo. Điều này đặc biệt quan trọng trong việc cải thiện độ chính xác của mô hình phân loại tin tức.

## 3.3 Huấn luyện mô hình SVM

Sau khi hoàn tất tiền xử lý và chuyển đổi văn bản sang vector TF-IDF, dữ liệu được chia thành tập huấn luyện và tập kiểm tra theo tỷ lệ 80%-20% với stratify theo nhãn để đảm bảo cân bằng các lớp. Mỗi văn bản được biểu diễn bằng vector TF-IDF gồm tối đa 5000 đặc trưng.

Mô hình được sử dụng là `LinearSVC` từ thư viện `sklearn`. Mục tiêu của SVM là tìm mặt phẳng tối ưu phân tách các lớp trong không gian đặc trưng. Trong quá trình huấn luyện, mô hình học trọng số $w$ và bias $b$ cho từng lớp dựa trên các vector đặc trưng TF-IDF. Quá trình huấn luyện được thực hiện trên tập huấn luyện với các tham số mặc định của `LinearSVC` và đảm bảo tính ngẫu nhiên được kiểm soát bằng `random_state=42`.

Sau khi huấn luyện, mô hình sẵn sàng dự đoán nhãn cho các văn bản trong tập kiểm tra.

🎯 Test Accuracy (SVM): 0.9118

📊 Classification Report:

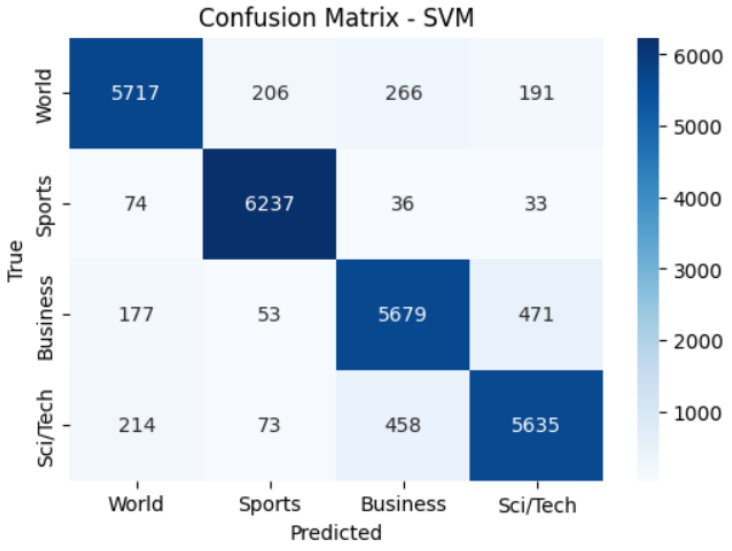
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| World | 0.92 | 0.90 | 0.91 | 6380 |
| Sports | 0.95 | 0.98 | 0.96 | 6380 |
| Business | 0.88 | 0.89 | 0.89 | 6380 |
| Sci/Tech | 0.89 | 0.88 | 0.89 | 6380 |
| accuracy | | | 0.91 | 25520 |
| macro avg | 0.91 | 0.91 | 0.91 | 25520 |
| weighted avg | 0.91 | 0.91 | 0.91 | 25520 |

Hình 1.1: Quá trình huấn luyện mô hình SVM. Mỗi vector văn bản được ánh xạ vào không gian đặc trưng TF-IDF, mô hình tìm mặt phẳng tối ưu phân tách các lớp.

## 3.4 Đánh giá mô hình SVM

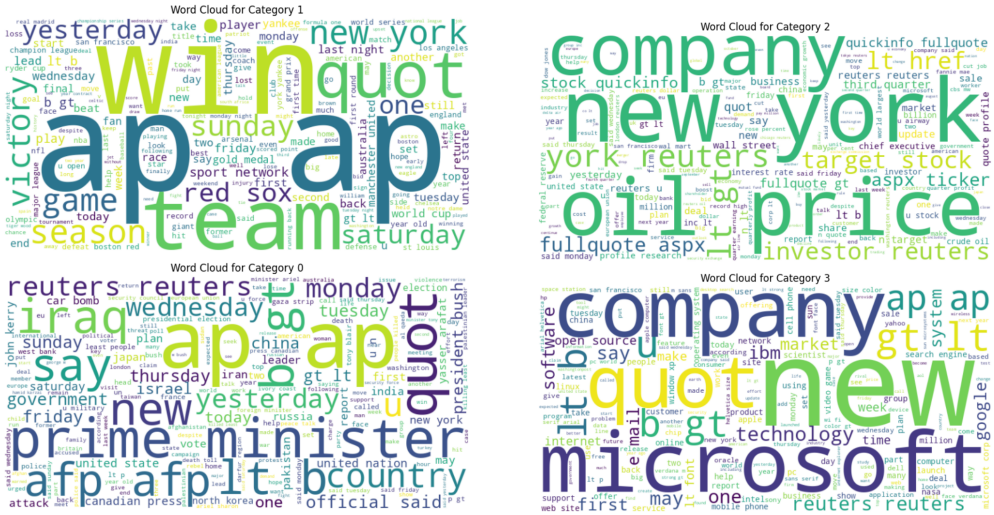
Mô hình SVM được đánh giá trên tập kiểm tra bằng độ chính xác (accuracy) và *classification report* bao gồm *precision*, *recall* và *F1-score*. Trên tập kiểm tra, độ chính xác đạt được là 0.92, cho thấy khả năng phân loại tốt các tin tức theo 4 lớp.

Để trực quan hóa hiệu quả phân loại, ma trận nhầm lẫn được tạo ra. Trong ma trận này, các hàng biểu diễn nhãn thực tế và các cột biểu diễn nhãn dự đoán. Các ô trên đường chéo thể hiện số lượng dự đoán đúng, trong khi các ô ngoài đường chéo thể hiện số lượng nhầm lẫn. Quan sát ma trận nhầm lẫn cho thấy lớp *World* và *Business* có một vài mẫu bị nhầm lẫn giữa nhau, trong khi lớp *Sports* và *Sci/Tech* được phân loại khá chính xác. Hiện tượng nhầm lẫn này xuất phát từ sự trùng lặp từ khóa giữa các lớp, ví dụ từ "market" có thể xuất hiện trong cả lớp *Business* và *Sci/Tech*.




Hình 1.2: Ma trận nhầm lẫn của mô hình SVM trên tập kiểm tra. Các ô trên đường chéo biểu thị số lượng dự đoán đúng, các ô ngoài đường chéo biểu thị nhầm lẫn giữa các lớp.

Ngoài ra, các *WordCloud* được tạo ra cho từng lớp, hiển thị trực quan các từ khóa quan trọng. Ví dụ, lớp *World* có các từ "president", "government", "war"; lớp *Sports* có "game", "team", "score"; lớp *Business* có "market", "company", "stock"; và lớp *Sci/Tech* có "technology", "software", "scientists". Các từ khóa này giúp giải thích trực quan tại sao mô hình có thể phân loại chính xác hoặc nhầm lẫn một số mẫu.




Hình 1.3: Ví dụ WordCloud cho lớp tin tức, thể hiện các từ xuất hiện nhiều với kích thước lớn hơn.

Tóm lại, việc kết hợp huấn luyện SVM trên đặc trưng TF-IDF với đánh giá bằng ma trận nhầm lẫn và WordCloud giúp cả việc đánh giá hiệu suất mô hình lẫn giải thích trực quan cách mô hình phân loại tin tức. Các từ khóa quan trọng trong WordCloud minh họa lý do một số mẫu được phân loại chính xác và lý do một số mẫu nhầm lẫn giữa các lớp có nội dung gần nhau.

## 3.5 Kết quả thực nghiệm


Hình 1.4: demo giao diện

Kết quả thực nghiệm của mô hình SVM trên tập dữ liệu AG-News được minh họa trong hình. Người dùng có thể nhập một đoạn văn bản tin tức và mô hình sẽ dự đoán nhãn tương ứng trong bốn nhóm: Thế giới, Thể thao, Kinh doanh, Khoa học/Công nghệ. Hình ảnh hiển thị giao diện cho thấy ô nhập văn bản bên trái, nơi người dùng nhập nội dung tin tức, và kết quả dự đoán hiển thị bên phải sau khi nhấn nút "Submit".

Thực nghiệm cho thấy mô hình SVM hoạt động hiệu quả với dữ liệu đã tiền xử lý, bao gồm làm sạch văn bản, chuẩn hóa chữ hoa/thường, loại bỏ stopwords và biến đổi thành ma trận TF-IDF. Nhờ vào quá trình huấn luyện trên tập dữ liệu lớn với nhiều từ khóa quan trọng, mô hình có khả năng nhận diện chính xác nhãn tin tức từ các đặc trưng văn bản. Kết quả trực quan trên hình minh họa tính năng phân loại trực tiếp và độ chính xác cao của mô hình trong việc dự đoán nhãn của từng đoạn tin tức.

Phần thực nghiệm này cho thấy rằng SVM, kết hợp với TF-IDF, là một lựa chọn hiệu quả cho bài toán phân loại văn bản, đồng thời cho phép xây dựng các ứng dụng dự đoán trực tiếp dựa trên văn bản nhập vào, tương tự như hệ thống thử nghiệm được minh họa.

# KẾT LUẬN

Trong báo cáo này, chúng em đã nghiên cứu và triển khai mô hình **Support Vector Machine (SVM)** cho bài toán phân loại tin tức, từ khâu tiền xử lý dữ liệu đến huấn luyện, đánh giá và phân tích kết quả. Quá trình thực nghiệm cho thấy SVM là một phương pháp mạnh mẽ và hiệu quả trong việc phân loại các văn bản tin tức theo chủ đề nhờ khả năng tìm kiếm ranh giới tối ưu giữa các lớp dữ liệu.

Các bước tiền xử lý, bao gồm loại bỏ stopwords, chuẩn hóa chữ, tách từ và vector hóa văn bản, đóng vai trò then chốt giúp mô hình học được các đặc trưng quan trọng và giảm nhiễu từ dữ liệu thô. Việc lựa chọn *kernel* phù hợp, đặc biệt là *linear kernel* cho dữ liệu text lớn, cùng với điều chỉnh tham số *C* giúp cân bằng giữa độ chính xác và khả năng tổng quát, đã góp phần nâng cao hiệu suất của mô hình. Kết quả đánh giá trên tập dữ liệu thực nghiệm cho thấy mô hình đạt độ chính xác cao, phù hợp với các ứng dụng thực tế như hệ thống gán nhãn tự động, lọc và phân loại tin tức theo chủ đề, hỗ trợ biên tập viên và cải thiện trải nghiệm người dùng.

Bên cạnh những ưu điểm, SVM cũng gặp một số hạn chế như tốn bộ nhớ và thời gian tính toán khi số lượng dữ liệu lớn, đồng thời không trực tiếp khai thác được ngữ nghĩa sâu của từ. Do đó, các cải tiến như áp dụng kỹ thuật giảm chiều dữ liệu (*PCA*) hoặc kết hợp với các mô hình học sâu có thể giúp nâng cao hiệu quả trên các tập dữ liệu thực tế phức tạp.

Tóm lại, SVM là một lựa chọn đáng tin cậy cho bài toán phân loại tin tức, đặc biệt khi dữ liệu được xử lý tốt và tham số được tối ưu hợp lý. Với việc tiếp tục cải thiện các bước tiền xử lý và kết hợp các kỹ thuật bổ trợ, hiệu suất của hệ thống có thể được nâng cao hơn nữa, mở ra khả năng ứng dụng rộng rãi trong các hệ thống phân loại văn bản tự động.

# TÀI LIỆU THAM KHẢO


[1] Aman Anandrai, *AG News Dataset*, https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset, truy cập ngày 09/09/2025.

[2] C. Cortes và V. Vapnik, *Support-Vector Networks*, Machine Learning, 20:273–297, 1995.

[3] https://machinelearningcoban.com/2017/04/09/smv/.

[4] https://docs.google.com/document/d/1gTwzLOab6PDMoJ0lKGJvC3s4ZBwrLWho/edit#heading=h.2s8eyo1