**BÁO CÁO TUẦN 3**

**Các thuật toán phân loại văn bản**

Có rất nhiều thuật toán phân loại văn bản như Naive Bayes Classifier, Decision Tree (Random Forest), Vector Support Machine (SVM), Boosting and Bagging algorithms, Convolution Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM, Bi-LSTM), ….

**Thuật toán NBC (Naive Bayes Classification):**

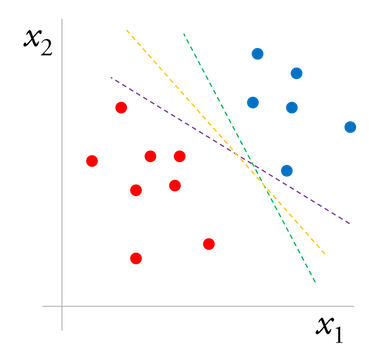
Naive Bayes Classification (NBC) là một thuật toán dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các phán đoán cũng như phân loại dữ liệu dựa trên các dữ liệu được quan sát và thống kê. Naive Bayes Classification là một trong những thuật toán được ứng dụng rất nhiều trong các lĩnh vực Machine learning dùng để đưa các dự đoán chính xác nhất dự trên một tập dữ liệu đã được thu thập, vì nó khá dễ hiểu và độ chính xác cao. Nó thuộc vào nhóm Supervised Machine Learning Algorithms (thuật toán học có hướng dẫn), tức là máy học từ các ví dụ từ các mẫu dữ liệu đã có.

Ví dụ như có thể ứng dụng vào việc thiết kế một ứng dụng nghe nhạc có thể phán đoán được sở thích của nghe nhạc của người dùng dựa trên các hành vi như nhấn nút “thích” bài hát, “nghe đi nghe” lại nhiều lần các bài hát, “bỏ qua” các bài hát không thích …. Dựa trên tập dữ liệu đó ta có thể áp dụng NBC để tính toán ra các phong cách nhạc mà người dùng thích nhất, từ đó chúng ta có thể đưa ra các “gợi ý” nghe nhạc gần đúng nhất cho người dùng từ việc học hỏi từ những thói quen đó.

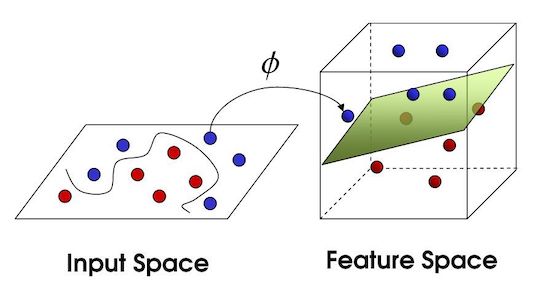
**Thuật toán SVM (Support Vector machines):**

Support Vector Machine (SVM) là một thuật toán thuộc nhóm Supervised Learning (Học có giám sát) dùng để phân chia dữ liệu (Classification) thành các nhóm riêng biệt.

Hình dung ta có bộ data gồm các điểm xanh và đỏ đặt trên cùng một mặt phẳng. Ta có thể tìm được đường thẳng để phân chia riêng biệt các bộ điểm xanh và đỏ như hình bên dưới.

Với những bộ data phức tạp hơn mà không thể tìm được đường thẳng để phân chia thì sao?

Ta cần dùng thuật toán để ánh xạ bộ data đó vào không gian nhiều chiều hơn (n chiều), từ đó tìm ra siêu mặt phẳng (hyperplane) để phân chia. Ví dụ trong hình bên dưới là việc ánh xạ tập data từ không gian 2 chiều sang không gian 3 chiều.

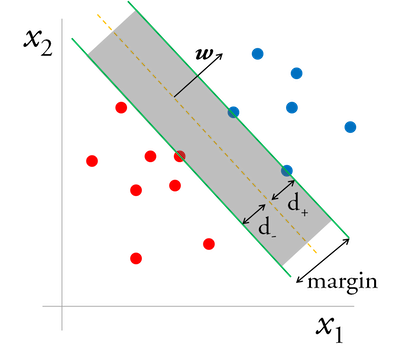
**Tối ưu thuật toán trong SVM**

Quay lại bài toán với không gian 2 chiều. Ở ví dụ trong hình đầu tiên, có thể tìm được rất nhiều các đường thẳng để phân chia 2 bộ điểm xanh, đỏ. Vậy đường thẳng như thế nào được coi là tối ưu?

Nhìn bằng mắt thường có thể thấy, đường tối ưu là đường tạo cho ta có cảm giác 2 lớp dữ liệu nằm cách xa nhau và cách xa đường đó nhất.

Tuy nhiên tính toán sự tối ưu bằng toán học, trong SVM sử dụng thuật ngữ Margin.

Margin là khoảng cách giữa siêu phẳng (trong trường hợp không gian 2 chiều là đường thẳng) đến 2 điểm dữ liệu gần nhất tương ứng với 2 phân lớp.

SVM cố gắng tối ưu thuật toán bằng các tìm cách maximize giá trị margin này, từ đó tìm ra siêu phẳng đẹp nhất để phân 2 lớp dữ liệu.

Bài toán của trở thành tìm ra 2 đường biên của 2 lớp dữ liệu (ở hình bên trên là 2 đường xanh lá cây) sao cho khoảng cách giữa 2 đường này là lớn nhất.

Đường biên của lớp xanh sẽ đi qua một (hoặc một vài) điểm xanh.

Đường biên của lớp đỏ sẽ đi qua một (hoặc một vài) điểm đỏ.

Các điểm xanh, đỏ nằm trên 2 đường biên được gọi là các support vector, vì chúng có nhiệm vụ support để tìm ra siêu phẳng. Đó cũng là lý do của tên gọi thuật toán Support Vector Machine.

**Ưu điểm của SVM**

Là một kĩ thuật phân lớp khá phổ biến, SVM thể hiện được nhiều ưu điểm trong số đó có việc tính toán hiệu quả trên các tập dữ liệu lớn. Có thể kể thêm một số ưu điểm của phương pháp này như:

* Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn
* Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết dịnh
* Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

**Nhược điểm của SVM**

* Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tồi
* Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm margin từ điểm dữ liệu mới đến siêu phẳng phân lớp.

**Thuật toán CNN (Convolution Neural Network):**

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

Hầu hết các mạng CNN đều được thiết kế theo nguyên tắc chung:

* Sử dụng nhiều convolution layer chồng lên nhau
* Giảm dần kích thước output mỗi tầng
* Tăng dần số lượng feature map

Trong khi các mạng CNN trước đây hầu hết theo dạng chỉ đơn giản thiết kế theo lối nhiều lớp Convolution xếp chồng lên nhau thì nhiều mạng mới đây đã thiết kế sáng tạo hơn và cho kết quả hiệu quả hơn. Các kiến trúc mạng này đóng vai trò rất quan trọng, nên học và áp dụng thành thạo đối những học viên ML cho những nhiệm vụ cụ thể trong Computer Vision. Các mạng này được dùng nhiều trong việc tách các feature để phục vụ cho các nhiệm vụ chuyên biệt. (mình sẽ có 1 bài về transfer learning và fine turning)

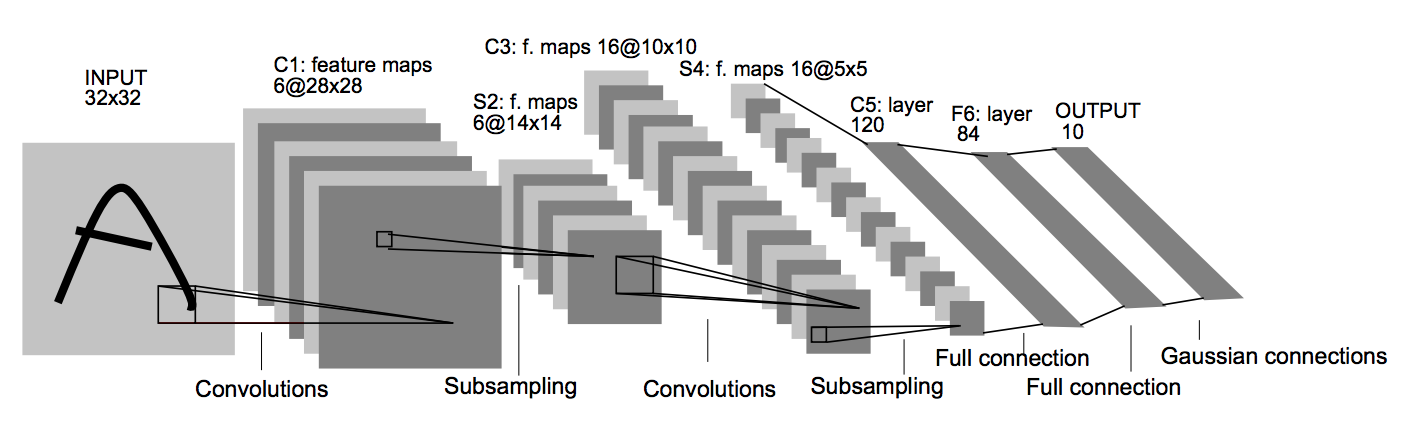
**Mạng truyền thống**

**LeNet-5**

Được tạo ra bởi Yann Lecun năm 1998, để nhận diện chữ viết tay. Mô hình này là cơ sở thiết kế cho các mạng sau này.

Cấu trúc như hình vẽ

Tham số: 60.000

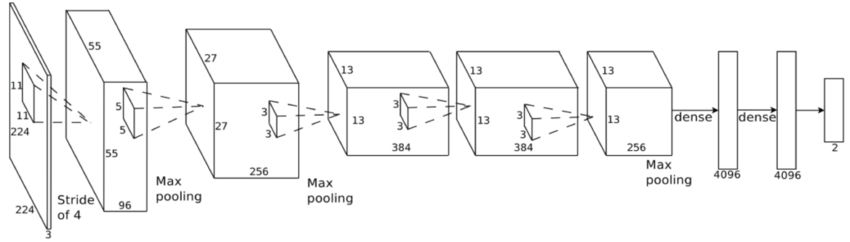
[[](https://www.jeremyjordan.me/content/images/2018/04/Screen-Shot-2018-04-16-at-11.34.51-AM.png)](https://www.jeremyjordan.me/content/images/2018/04/Screen-Shot-2018-04-16-at-11.34.51-AM.png" \o "Screen-Shot-2018-04-16-at-11.34.51-AM.png)

**[Screen-Shot-2018-04-16-at-11.34.51-AM.png1414×432](https://www.jeremyjordan.me/content/images/2018/04/Screen-Shot-2018-04-16-at-11.34.51-AM.png" \o "Screen-Shot-2018-04-16-at-11.34.51-AM.png)**

**AlexNet**

Được phát triển bởi Alex Krizhevsky năm 2012 trong cuộc thi ImageNet 2012

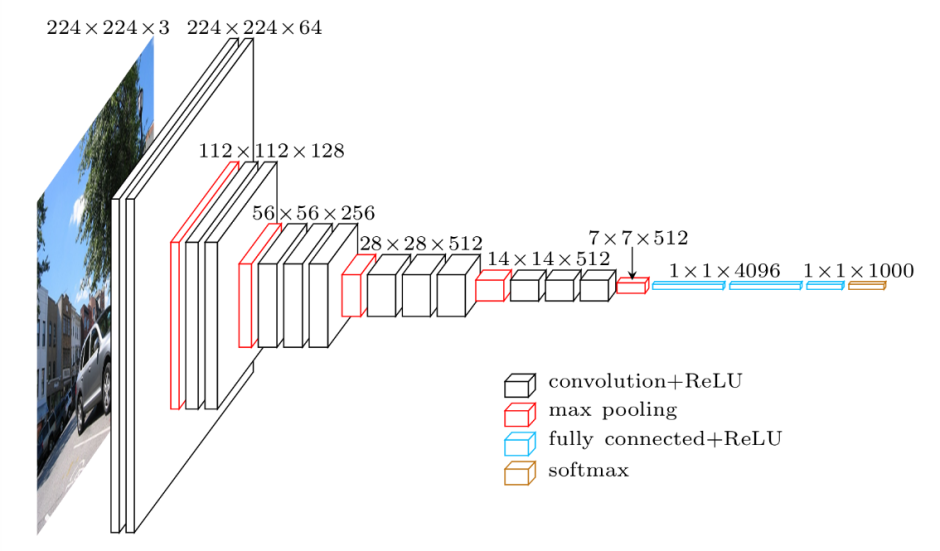
Kiến trúc tương đối giống với LeNet-5. Khác biệt ở điểm mạng này được thiết kế lớn hơn, rộng hơn Lượng Tham số: 60.000.000 (gấp 1000 lần LeNet-5 ) Kiến trúc như hình dưới đây:

[[](https://www.jeremyjordan.me/content/images/2018/04/AlexNet-CNN-architecture-layers.png)](https://www.jeremyjordan.me/content/images/2018/04/AlexNet-CNN-architecture-layers.png" \o "AlexNet-CNN-architecture-layers.png)

**[AlexNet-CNN-architecture-layers.png850×242](https://www.jeremyjordan.me/content/images/2018/04/AlexNet-CNN-architecture-layers.png" \o "AlexNet-CNN-architecture-layers.png)**

**VGG-16**

Được phát triển năm 2014, là một biến thể sâu hơn nhưng lại đơn giản hơn so với kiến trúc convolution (từ gốc: convolutional structure) thường thấy ở CNN Kiến trúc như hình dưới, có thể thấy số mặc dù các tầng cao hơn được đơn giản hóa so với LeNet, AlexNet … thu gọn về kích thước nhưng số lượng lại lớn hơn, sâu hơn  
Số lượng tham số: 138.000.000

[[](https://www.jeremyjordan.me/content/images/2018/04/vgg16.png)](https://www.jeremyjordan.me/content/images/2018/04/vgg16.png" \o "vgg16.png)

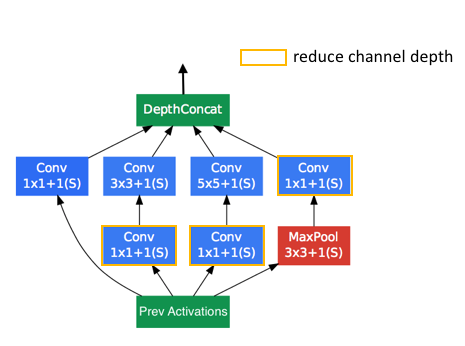
**[vgg16.png940×551](https://www.jeremyjordan.me/content/images/2018/04/vgg16.png" \o "vgg16.png)**

**Inception (GoogleLeNet)**

Vào năm 2014, các nhà nghiên cứu của google đã đưa ra mạng Inception -> tham dự cuộc thi ImageNet 2014.

Mô hình này khá đặc biệt, không hoàn toàn là các tầng layer nối tiếp gối đầu lên nhau như các mạng trên. Mạng gồm các đơn vị gọi là “inception cell” (cái này mình nghĩ không nên dịch) - thực hiện convolution 1 input với nhiều filter khác nhau rồi tổng hợp lại, theo nhiều nhánh (branch).

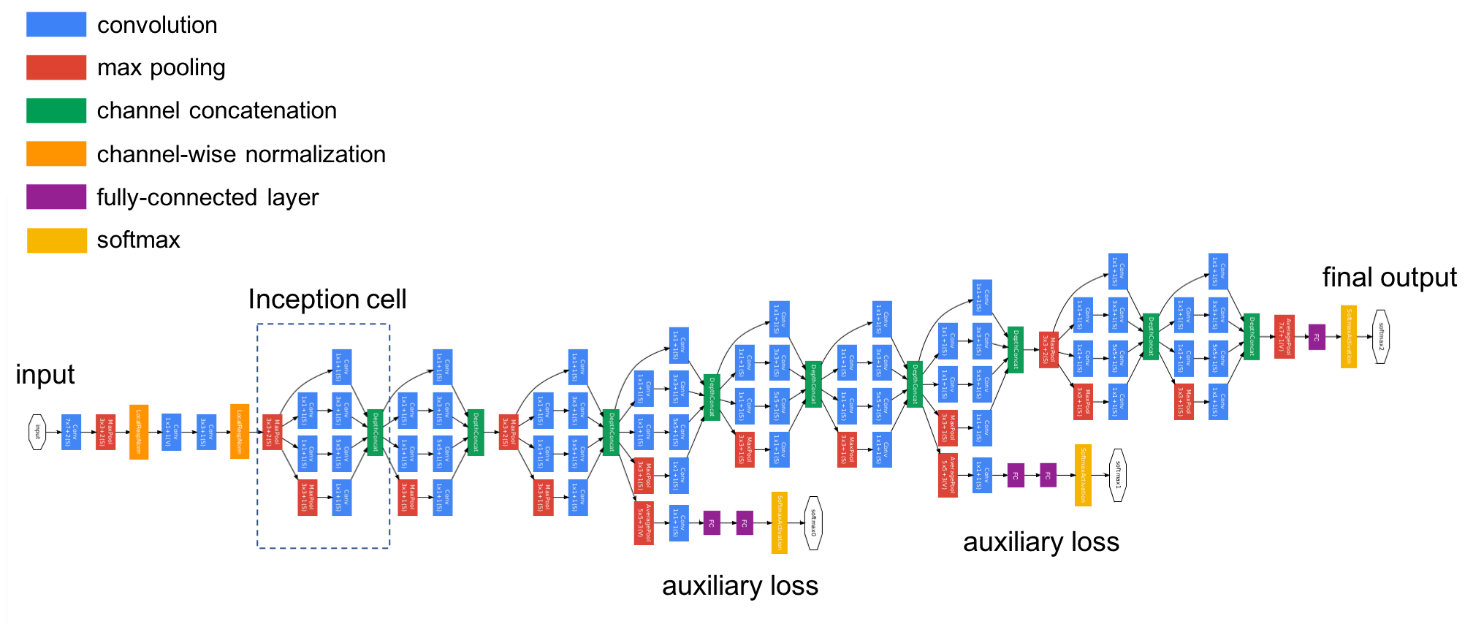
Để tiết kiệm tính toán, các convolution kích thước 1x1 được sử dụng để giảm chiều sâu chanel của input (reduce the input channel depth). Với mỗi 1 cell, sử dụng các 1x1, 3x3, 5x5 filter để extract các feature từ input.  
Dưới đây là dạng 1 cell.



Các nhà nghiên cứu đã thử thay các filter 3x3, 5x5 bằng các filter lớn hơn như 7x7, 11x11 . Có hiệu quả hơn về việc extract các feature, tuy nhiên đánh đổi lại là thời gian tính toán lâu hơn rất nhiều. Người ta cũng nhận ra rằng 1 filter 5x5 có thể thay bằng 2 filter 3x3 nối nhau, hiệu quả tương đương nhưng tiết kiệm được tính toán rất lớn.

Dưới đây là kiến trúc mạng Inception

Mạng được xây dựng từ việc ghép các inception cell lại với nhau.

[[](https://www.jeremyjordan.me/content/images/2018/04/inception-model.png)](https://www.jeremyjordan.me/content/images/2018/04/inception-model.png" \o "inception-model.png)

**[inception-model.png2016×854](https://www.jeremyjordan.me/content/images/2018/04/inception-model.png" \o "inception-model.png)**

Inception có một đặc điểm khá hay là có thêm 2 output phụ. Người ta tin rằng hai output phụ này không quá ảnh hưởng tới chất lượng của mạng trong khi train những epoch đầu. Nó giúp cho việc train diễn ra nhanh hơn khi tối ưu những layer đầu dựa vào các output phụ ( trong những epoch đầu). Có thể nghĩ đơn giản rằng trong những epoch đầu, các layer càng gần cuối càng ít được tối ưu ngay => chưa cần thiết phải tối ưu ngay. Sau một thời gian tối ưu các layer đầu rồi mới tối ưu các layer tiếp theo dựa vào final input. Việc này cải thiện khả năng tính toán và tốc độ train khá nhiều.