Chương 2: TIẾN TRÌNH HỌC MÁY

**5 bước thực hiện một tác vụ máy học:**

* **Collecting data (thu thập dữ liệu):** thu thập các dạng dữ liệu cần dùng cho máy học: văn bản, sensor, hình ảnh, âm thanh, …
* **Preparing data (chuẩn bị dữ liệu):** chọn những dữ liệu có chất lượng, loại bỏ dữ liệu cá biệt (outlier)
* **Training a model (học mô hình/huấn luyện mô hình):** chia dữ liệu thu thập thành tập học (training) và tập thử (testing), học một mô hình máy học.
* **Evaluating a model (đánh giá mô hình)**: đánh giá độ chính xác dự đoán (accuracy prediction) của mô hình được học.
* **Improving the performance (cải tiến sự thể hiện)**: chọn mô hình học khác, sử dụng thêm biến (variable) học.

**Tiền xử lý dữ liệu**

* Dữ liệu là tập hợp các đối tượng (object) và các thuộc tính (attributes) của chúng
* Thuộc tính: đặc điểm của một đối tượng.
* VD: màu mắt của một người, nhiệt độ của một địa điểm tại một thời điểm xác định
* Tập các thuộc tính mô tả một đối tượng

**Một số loại dữ liệu:**

* **Categorical - Kiểu liệt kê**
* Dữ liệu là một tập xác định các giá trị
* Ví dụ: màu mắt (xanh, nâu, đen); xếp hạng (tốt, trung bình, xấu); chiều cao (cao, trung bình, thấp)
* Nhị phân - Binary (Yes/No, True/False).
* **Numeric - Kiểu số**
  + Dữ liệu nhiệt độ, thời gian (ngày, giờ, năm); chiều dài; chiều cao
  + Rời rạc - Discrete (7 ngày trong tuần)
  + Liên tục - Continuous (370, 37.50, 37.30, …)
  + Nhị phân - Binary (0/1).
* **Dữ liệu văn bản**

Mỗi văn bản là một vector chứa các từ/ thuật ngữ (term), giá trị của mỗi thuộc tính chính là số lần xuất hiện của các từ/ thuật ngữ trong văn bản.

* **Dữ liệu giao dịch**
  + Mỗi giao dịch chứa tập hợp các mục dữ liệu (item).
  + Tập các mục dữ liệu được biểu diễn dưới dạng binary vector.
* **Dữ liệu có thứ tự (Ordered Data)**
  + Dữ liệu gen (Genomic sequence data): dữ liệu là chuỗi có thứ tự các ký tự.
  + Dữ liệu chuỗi thời gian (Time series).
* **Dữ liệu đồ thị**
  + Web graph and HTML links
* **Không gian, ảnh và đa phương tiện**
  + Dữ liệu không gian: bản đồ GIS (raster, vector)
  + Dữ liệu ảnh
  + Dữ liệu video, ...

**Tại sao phải tiền xử lý dữ liệu?**

**No quality data, no quality mining results!**

* + Để đưa ra quyết định hiệu quả cần phải dựa trên dữ liệu chất lượng
  + Kho dữ liệu cần phải được tích hợp bởi các dữ liệu chất lượng
  + Trùng lắp hay thiếu dữ liệu sẽ dẫn đến việc thống kê sai hay hiểu nhầm đặc điểm của dữ liệu

**Data preparation, cleaning and transformation comprises the majority of the work in a data mining application (90%)**

**“Data in the real world is dirty”**

* + *Không đầy đủ (Imcomplete*): thiếu giá trị của thuộc tính, thiếu thông tin thuộc tính quan tâm, … VD: nghề nghiệp = “ “
  + *Nhiễu (Noisy)*: chứa sai sót hoặc ngoại lệ

**D.Lương = “-40”**

* *Không nhất quán (Inconsistent)*: có sự sai biệt trong mã hoặc tên
* Ví dụ đánh giá “1,2,3” và đánh giá “A,B,C”
* Ví dụ sự khác biệt giữa bản ghi trùng lặp

**Công việc/ nhiệm vụ của Tiền xử lý dữ liệu**

* Làm sạch
* Tích hợp
* Chuyển đổi
* Giảm thiểu
* **Data cleaning - làm sạch dữ liệu**
  + Điền các giá trị còn thiếu, làm trơn các dữ liệu nhiễu (smoth noisy data), xác định hay loại bỏ các ngoại lệ, giải quyết dữ liệu không nhất quán
* **Data integration - tích hợp dữ liệu**
  + Tích hợp nhiều cơ sở dữ liệu, khối dữ liệu, tệp tin hoặc ghi chú
* **Data transformation - chuyển đổi**
  + Chuẩn hóa dữ liệu (scaling to a special range) - Naive Bayes
  + Kết hợp dữ liệu (Aggegation)
* **Data reduction - thu giảm dữ liệu**
  + Thu được sự thể hiện giảm về thể tích nhưng tạo ra kết quả phân tích tương tự hoặc tương tự
  + Sự rời rạc hóa dữ liệu: có tầm quan trọng đặc biệt, đặc biệt đối với dữ liệu số
  + Tổng hợp dữ liệu, giảm kích thước, nén dữ liệu, khái quát hóa

**Data cleaning - làm sạch dữ liệu**

* Fill in missing values - bổ sung dữ liệu bị thiếu
* Identify outliers and smooth out noisy data: nhận diện phần tử biên và giảm thiểu nhiễu
* Correct inconsistent data: xử lý dữ liệu không nhất quán
* **Fill in missing values - bổ sung dữ liệu bị thiếu**
* Data is not always available
  + Nguyên nhân gây thiếu dữ liệu
    - Sự cố thiết bị
    - Không tương thích với dữ liệu trước đó nên giá trị (mới) bị xóa đi
    - Dữ liệu không được nhập vào (lỗi người nhập liệu)
    - Không lưu trữ lịch sử hay sự thay đổi của dữ liệu (thông tin thuyên chuyển cán bộ trong 1 đơn vị)
* Cách xử lý dữ liệu bị thiếu
* Bỏ qua các bản ghi có thuộc tính bị thiếu
* Bổ sung dữ liệu thiếu bằng tay
* Bổ sung dữ liệu thiếu tự động
* Giá trị trung bình của thuộc tính, của thuộc tính của cùng lớp
* Giá trị hằng số nhất định
* Giá trị có thể xảy ra nhất
* **Identify outliers and smooth out noisy data - xử lý dữ liệu bị nhiễu**
  + Dữ liệu nhiễu là dữ liệu (đối tượng) không tuân theo đặc tính/ hành vi chung của tập dữ liệu
  + Giá trị không chính xác do
    - Lỗi do thiết bị thu thập dữ liệu
    - Vấn đề nhập dữ liệu: người dùng hoặc máy có thể sai
    - Vấn đề truyền dữ liệu: sai từ thiết bị gửi/ nhận/ truyền
    - Hạn chế của công nghệ: ví dụ phần mềm có thể xử lý không đúng
    - Thiếu nhất quán khi đặt tên: cũng một tên nhưng cách viết khác nhau
  + Phân khoảng (Binning)
    - Sắp dữ liệu tăng và chia “đều” vào các thùng (bin)
    - Làm trơn: theo trung bình, theo trung vị
    - Hồi quy (Regression): gắn dữ liệu với một hàm hồi quy (Regression function)
  + Phân cụm (Clustering): phát hiện và loại bỏ các ngoại lai (sau khi đã xác định các cụm)
  + Kết hợp giữa máy tính (phát hiện) và kiểm tra của con người (hiệu chỉnh)
* **Phương pháp phân khoảng làm trơn dữ liệu (Binning methods for data smoothing)**
  + Dữ liệu được xếp theo giá trị: 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
  + Phân chia theo chiều sâu
    - Bin 1: 4, 8, 9, 15
    - Bin 2: 21, 21, 24, 25
    - Bin 3: 26, 28, 29, 34
  + Làm trơn theo trung bình
    - Bin 1: 9, 9, 9, 9
    - Bin 2: 23, 23, 23, 23
    - Bin 3: 29, 29, 29, 29
  + Làm trơn bin theo biên
    - Bin 1: 4, 4, 4, 15
    - Bin 2: 21, 21, 21, 25
* Bin 3: 26, 26, 26 34
* **Correct inconsistent data: xử lý dữ liệu không nhất quán**
  + Dữ liệu ghi nhận khác nhau (9/3/2018 và 3/9/2018)
  + Nguyên nhân gây ra sự không nhất quán:
    - Sự không nhất quán trong các qui ước đặt tên hay mã dữ liệu
    - Định dạng không nhất quán của các vùng nhập liệu
    - Thiết bị ghi nhận dữ lieu
  + Giải pháp
    - Tạo các ràng buộc khi nhập liệu
    - Điều chỉnh dữ liệu không nhất quán bằng tay sau khi nhập liệu
    - Viết các giải thuật điều chỉnh, chuyển đổi tự động

**Data integration - tích hợp dữ liệu**

* **Tích hợp dữ liệu (Data integration)**
  + Kết hợp dữ liệu từ nhiều nguồn vào một kho dữ liệu thống nhất
* **Tích hợp ở mức mô hình (Schema integration)**
  + Tích hợp metadata từ các nguồn khác nhau
* **Vấn đề xác định thực thể (để tránh dư thừa dữ liệu)**
  + Cần xác định các thực thể (identities) trên thực tế từ nhiều nguồn dữ liệu
* **Phát hiện và xử lý các mâu thuẫn đối với giá trị dữ liệu**
  + Đối với cùng một thực thể trên thực tế, nhưng các giá trị thuộc tính từ nhiều nguồn khác nhau lại khác nhau. Các lý do có thể:
    - Các cách biểu diễn khác nhau
* Mức đánh giá, độ đo (scale) khác nhau. VD: độ đo m và inch.
* **Dư thừa dữ liệu (Redundant data) thường xuyên xảy ra, khi tích hợp dữ liệu từ nhiều nguồn**
  + Định danh đối tượng: cùng một thuộc tính (hay cùng một đối tượng) có thể mang các tên (định danh) khác nhau trong các csdl khác nhau
  + Dữ liệu suy ra được: một thuộc tính trong một bảng có thể là một thuộc tính được suy ra (derived attribute) trong một bảng khác
* **Các thuộc tính dư thừa có thể được phát hiện bằng phân tích tương quan (Correlation analysis)**
* **Yêu cầu chung đối với quá trình tích hợp dữ liệu**: giảm thiểu (tránh được là tốt nhất) các dư thừa và các mâu thuẫn 
  + Giúp cải thiện tốc độ của quá trình khai phá dữ liệu và nâng cao chất lượng của các kết quả (tri thức) thu được

**Data transformation - chuyển đổi dữ liệu**

* **Biến đổi dữ liệu**: quá trình biến đổi hay kết hợp dữ liệu vào những dạng thích hợp cho quá trình khai phá dữ liệu
* **Làm trơn dữ liệu (Smoothing):** ước lượng Laplace
* **Kết hợp dữ liệu (Aggregation)**
* **Chuẩn hóa (Normolization)**
  + Thuộc tính X2 có miền giá trị (0 … 100) trong khi thuộc tính X1 có miền giá trị (0 … 1)
  + Kết quả phụ thuộc nhiều vào X2 (chênh lệch X2 lớn hơn so với X1)
  + Nên chuẩn hóa dữ liệu (chuẩn hóa X2 về giá trị 0 … 1)

**new\_val = (val – min)(max - min)**

**Data reduction - thu giảm dữ liệu**

* **Kho dữ liệu chứa tới hàng TB**
  + Phân tích/ khai phá dữ liệu phức mất thời gian rất dài khi chạy trên toàn bộ dữ liệu
* **Rút gọn/ thu giảm dữ liệu**
  + Có được trình bày gọn của tập dữ liệu mà nhỏ hơn nhiều về khối lượng mà sinh ra cùng (hoặc hầu như cùng) kết quả
* **Chiến lượng rút gọn dữ liệu**
  + Rút gọn đặc trưng - loại bỏ thuộc tính không quan trọng
  + Kết hợp khối dữ liệu
  + Thu giảm chiều: PCA (phân tích thành phần chính)

**Lựa chọn mô hình máy học**

Phân loại các trang web theo chủ đề

* Xác định T ?
* Xác định P ?
* Xác định E ?
* Xác định T → Phân loại các trang Web theo các chủ đề đã định trước
* Xác định P → Tỷ lệ (%) các trang Web được phân loại chính xác
* Xác định E → Một tập các trang web, trong đó mỗi trang web gắn với một chủ đề

Nhận dạng chữ viết tay

* Xác định T ?
* Xác định P ?
* Xác định E ?
* Xác định T → Nhận dạng và phân loại các từ trong các ảnh chữ viết tay
* Xác định P → Tỷ lệ (%) các từ được nhận dạng và phân loại đúng
* Xác định E → Một tập các ảnh chữ vết tay, trong đó mỗi ảnh được gắn với một định danh của một từ
* Việc lựa chọn mô hình cần tìm ra sự thỏa hiệp (compromise) phù hợp giữa
  + Mức độ phức tạp của mô hình hệ thống học được
  + Mức độ chính xác về dự đoán của hệ thống đối với tập huấn luyện
* Nguyên lý Occam’s razor: một mô hình tốt là một mô hình đơn giản đạt độ chính xác (về phân loại/dự đoán) cao đối với tập dữ liệu được sử dụng.

**Đánh giá hiệu năng hệ thống học máy**

* Trong thực tế, một bài toán Machine Learning có thể được giải quyết bởi nhiều phương pháp, cho ra những mô hình khác nhau.
* Đứng trước nhiều sự lựa chọn, làm cách nào để chúng ta có thể chọn ra mô hình phù hợp nhất cho bài toán đang giải quyết?
* → Cần phải đánh giá “hiệu năng” của mô hình trên dữ liệu mới

Đánh giá mô hình giúp ta trả lời những câu hỏi sau:

* Mô hình đã được huấn luyện thành công hay chưa?
* Mức độ thành công của mô hình tốt đến đâu?
* Khi nào nên dừng quá trình huấn luyện?
* Khi nào nên cập nhật mô hình?
* Việc **đánh giá mô hình** thường được thực hiện trên dữ liệu mà mô hình chưa từng được học – trên **validation set và test set**
* Những bài toán khác nhau sẽ có những tiêu chí đánh giá khác nhau, vì vậy cần phải xác định rõ thứ tự ưu tiên của các tiêu chí cho việc đánh giá mô hình.
* Bên cạnh đó, đối với những tiêu chí phụ thuộc vào nhau, tức nếu cải thiện “hiệu năng” ở tiêu chí A thì tiêu chí B sẽ có “hiệu năng” thấp, cần phải đánh đổi (tradeoff) giữa các tiêu chí, thì chúng ta cần sử dụng các độ đo (metric) nhằm thuận tiện hơn cho việc đánh giá mô hình
* Các **phương pháp đánh giá (evaluation methods**)
  + Làm sao có được một đánh giá đáng tin cậy về hiệu năng của hệ thống
* Các **tiêu chí đánh giá (evaluation metrics)**
  + Làm sao để đo (tính toán) hiệu năng của hệ thống

**Các phương pháp đánh giá**

* Làm thế nào để thu được một đánh giá đáng tin cậy về hiệu năng của hệ thống?
  + Tập huấn luyện càng lớn, thì hiệu năng của hệ thống học càng tốt
  + Tập kiểm thử càng lớn, thì việc đánh giá càng chính xác
  + Vấn đề: Rất khó (ít khi) có thể có được các tập dữ liệu (rất) lớn
* Hiệu năng của hệ thống không chỉ phụ thuộc vào giải thuật học máy được sử dụng, mà còn phụ thuộc vào:
  + Phân bố lớp (Class distribution)
  + Chi phí của việc phân lớp sai (Cost of misclassification)
  + Kích thước của tập kiểm thử (Size of the test set)

**Các phương pháp:**

* Hold-out
* Straified sampling
* Repeated hold-out
* Cross-validation
  + K-fold
  + Leave-one-out

**Hold-out (Splitting)**

* Toàn bộ tập dữ liệu D được chia thành 2 tập con **không giao nhau**
  + Tập huấn luyện D\_train để huấn luyện hệ thống
  + Tập kiểm thử D\_test – để đánh giá hiệu năng của hệ thống đã học
* Các yêu cầu
  + Bất kỳ dữ liệu nào thuộc vào tập kiểm thử D\_test đều không được sử dụng trong quá trình huấn luyện hệ thống
  + Bất kỳ dữ liệu nào được sử dụng trong giai đoạn huấn luyện hệ thống đều không được sử dụng trong giai đoạn đánh giá hệ thống
  + Các dữ liệu kiểm thử trong D\_test cho phép một đánh giá không thiên vị đối với hiệu năng của hệ thống
* Các lựa chọn thường gặp: |D\_train|=(2/3).|D|, |D\_test|=(1/3).|D|
* Phù hợp khi ta có tập dữ liệu D có kích thước lớn

**Stratified sampling**

* Đối với các tập dữ liệu có kích thước nhỏ hoặc không cân xứng (Unbalanced datasets), các dữ liệu trong tập huấn luyện và thử nghiệm có thể không phải là đại diện
* Ví dụ: có (rất) ít, hoặc không có, các dữ liệu đối với một số lớp
* Mục tiêu: phân bố lớp (Class distribution) trong tập huấn luyện và tập kiểm thử phải xấp xỉ như trong toàn bộ tập dữ liệu
* Lấy mẫu phân tầng (Stratified sampling)
  + Là một phương pháp để cân xứng (về phân bố lớp)
  + Đảm bảo tỷ lệ phân bố lớp (tỷ lệ dữ liệu giữa các lớp) trong tập huấn luyện và tập kiểm thử là xấp xỉ nhau
* Phương pháp lẫy mẫu phân tầng không áp dụng được cho bài toán học máy dự đoán/ hồi quy (vì giá trị đầu ra của hệ thống là một giá trị số, không phải là một nhãn lớp)

**Repeated hold-out**

* Áp dụng phương pháp đánh giá Hold-out nhiều lần, để sinh ra (sử dụng) các tập huấn luyện và thử nghiệm khác nhau
  + Trong mỗi bước lặp, một tỷ lệ nhất định của tập D được lựa chọn ngẫu nhiên để tạo nên tập huấn luyện (có thể sử dụng kết hợp với phương pháp lấy mẫu phân tầng - Stratified sampling)
  + Các giá trị lỗi (hoặc các giá trị đối với các tiêu chí đánh giá khác) ghi nhận được trong các bước lớp này được lấy trung bình cộng (Average) để xác định giá trị lỗi tổng thể
  + Phương pháp này vẫn không hoàn hảo
  + Mỗi bước lặp sử dụng một tập kiểm thử khác nhau
  + Có một số dữ liệu trùng lặp (được sử dụng lại nhiều lần) trong các tập kiểm thử này

**Cross-validation**

* Để tránh việc trùng lặp giữa các tập kiểm thử (một số dữ liệu cùng xuất hiện trong các tập kiểm thử khác nhau)
* k-fold cross-validation
  + Tập toàn bộ các dữ liệu được chia thành k tập con không giao nhau (gọi là “fold”) có kích thước xấp xỉ nhau
  + Mỗi lần (trong số k lần) lặp, một tập con được sử dụng làm tập kiểm thử và k-1 tập con còn lại được dùng làm tập huấn luyện
  + K giá trị lỗi (mỗi giá trị tương ứng với một fold) được tính trung bình cộng để thu được giá trị lỗi tổng thể
* Các lựa chọn thông thường của k: 10 hoặc 5
* Thông thường, mỗi tập con (fold) được lấy mẫu phân tầng (xấp xỉ phân bố lớp) trước khi áp dụng quá trình đánh giá Cross-validation
* Phù hợp khi ta có tập dữ liệu vừa và nhỏ

**Leave-one-out Cross-validation**

* Một trường hợp (kiểu) của phương pháp Cross-validation
  + Số lượng các nhóm (folds) bằng kích thước của tập dữ liệu
  + Mỗi nhóm (fold) chỉ bao gồm một dữ liệu
* Khai thác tối đa tập dữ liệu ban đầu
* Không có bước lấy mẫu ngẫu nhiên (no random sub-sampling)
* Áp dụng lấy mẫu phân tầng (Straification) không phù hợp
  + Vì ở mỗi bước lặp, tập thử nghiệm chỉ gồm có một dữ liệu
* Chi phí tính toán (rất) cao
* Phù hợp khi có tập dữ liệu (rất) nhỏ

**Boostrap sampling**

* Phương pháp Cross-validation sử dụng việc lấy mẫu không lặp lại (Sampling without replacement)
  + Đối với mỗi dữ liệu, một khi đã được chọn (được sử dung) thì không thể được chọn (sử dụng) lại cho tập huấn luyện
* Phù hợp với tập dữ liệu có kích thước (rất) nhỏ
* Phương pháp Bootstrap sampling sử dụng việc lấy mẫu có lặp lại (Sampling with replacement) để tạo nên tập huấn luyện
* Giả sử tập dữ liệu D gồm n mẫu dữ liệu
* Lấy mẫu có lặp lại n lần đối với tập D, để tạo nên tập huấn luyện D\_train gồm n mẫu dữ liệu
  + Từ tập D, lấy ra ngẫu nhiên một mẫu dữ liệu X (nhưng không loại bỏ X ra khỏi tập D)
  + Đưa mẫu dữ liệu X vào trong tập huấn luyện D\_train
  + Lặp lại 2 bước trên n lần
* Sử dụng tập D\_train để huấn luyện hệ thống
* Sử dụng tất các cả dữ liệu thuộc D nhưng không thuộc D\_train để tạo nên tập thử nghiệm D\_test

**Các tiêu chí đánh giá**

* **Tính chính xác (Accuracy)**
  + Mức độ dự đoán (phân lớp) chính xác của hệ thống (đã được huấn luyện) đối với các dữ liệu kiểm tra (Test instances)
* **Tính hiệu quả (Eficiency)**
  + Chi phí về thời gian và tài nguyên (bộ nhớ) cần thiết cho việc huấn luyện và kiểm thử hệ thống
* **Khả năng xử lý nhiễu (Robustness)**
  + Khả năng xử lý (chịu được) của hệ thống đối với các dữ liệu nhiễu (lỗi) hoặc thiếu giá trị.
* **Khả năng mở rộng (Scalability)**
  + Hiệu năng của hệ thống (tốc độ học/ phân loại) thay đổi như thế nào đối với kích thước của tập dữ liệu
* **Khả năng diễn giải (Interpretability)**
  + Mức độ dễ hiểu (đối với người sử dụng) của các kết quả và hoạt động của hệ thống
* **Mức độ phức tạp (Complexity)**
  + Mức độ phức tạp của mô hình hệ thống (hàm mục tiêu) học được

**Tính chính xác**

* **Đối với bài toán phân loại**
  + Giá trị (kết quả) đầu ra của hệ thống là một giá trị định danh

**A black and white symbol

Description automatically generated**

**A black math equation with numbers

Description automatically generated with medium confidence**

* + **Trong đó**
    - x: dữ liệu trong tập D\_test
    - o(x): giá trị đầu ra (phân lớp) bởi hệ thống đối với dữ liệu x
    - c(x): phân lớp thực sự (đúng) của dữ liệu x
* **Đối với bài toán hồi quy (dự đoán)**
  + Giá trị (kết quả) đầu ra của hệ thống là một giá trị số

A black text on a white background

Description automatically generated

A black math symbol with a white background

Description automatically generated

* + **Trong đó**
    - o(x): giá trị đầu ra (dự đoán) bỏi hệ thống đối với dữ liệu x
    - d(x): giá trị đầu ra thực sự (đúng) của dữ liệu x
    - Accuracy là một hàm đảo (inverse function) đối với Error

**Ma trận nhầm lẫn (Confusion Matrix)**

* Nhược điểm của Accuracy là chỉ cho ta biết độ chính xác khi dự báo của mô hình, nhưng không thể hiện mô hình đang dự đoán sai như thế nào, vì vậy chúng ta cần một phương pháp đánh giá khác – Confusion Matrix
* Còn được gọi là Contingency Table
* Chỉ được sử dụng đối với bài toán phân loại (không thể áp dụng cho bài toán hồi quy - dự đoán)
* Là một ma trận thể hiện số lượng điểm dữ liệu thuộc vào một lớp và được dự đoán thuộc vào lớp
* Cung cấp thêm thông tin về tỉ lệ phân lớp đúng giữa các lớp, hay giúp phát hiện các lớp có tỉ lệ phân lớp nhầm cao nhờ vào các khái niệm True (False) Positive (Negative)

A diagram of negative and negative

Description automatically generated

* **True Positive (TP):** đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng)
* **True Negative (TN):** đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng)
* **False Positive (FP):** đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai) – Type I Error
* **False Negative (FN):** đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai) – Type II Error
* **Ví dụ:** cần dự đoán kết quả xét nghiệm COVID-19 của 1000 người nghi nhiễm. Dưới đây là kết quả dự đoán của mô hình:
  + Mô hình dự đoán có 30 ca dương tính, trong khi thực tế có 13 người nhiễm COVID-19
  + Mô hình dự đoán có 970 ca âm tính, nhưng thực tế trong 970 ca đó có 20 ca dương tính
* Biểu diễn kết quả dự đoán của mô hình bằng confusion matrix như sau:

**A white rectangular table with black text

Description automatically generated**

* Biểu diễn kết quả dự đoán của mô hình bằng confusion matrix như sau
  + True Positive TP = 13: có 13 người nhiễm COVID-19 được mô hình dự đoán đúng
  + False Positive FP = 17: có 17 người âm tính với COVID-19, nhưng được mô hình dự đoán dương tính
  + True Negative TN = 950: 950 trường hợp âm tính được mô hình phân loại chính xác
  + False Negative FN = 20: có 20 trường hợp dương tính với COVID-19 nhưng bị mô hình phân loại sai
* Vào ngày 16/04/2021 tại bệnh viện A có 100 bệnh nhân đến khám một loại bệnh, giả sử biết trước trong 100 bệnh nhân có 60 người mắc bệnh, 40 người không có bệnh. Sau khi thăm khám, bệnh viện đưa ra kết quả:
  + Trong 60 người bệnh thật thì có 45 người chuẩn đoán có bệnh, 15 người chuẩn đoán không mắc bệnh.
  + Trong 40 người không mắc bệnh thì có 30 người chuẩn đoán không mắc bệnh, 10 người chuẩn đoán là mắc bệnh.

A screenshot of a diagram

Description automatically generated

* **Bài tập:** dự đoán kết quả xét nghiệm của 1005 bệnh nhân xem họ có bị ung thư hay không. Dưới đây là những gì mô hình dự đoán:
  + 90 bệnh nhân bị ung thư và tất cả dự đoán đều đúng.
  + 915 bệnh nhân không bị ung thư nhưng thật ra có tới 910 người bị ung thư trong thực tế.

**A screenshot of a computer

Description automatically generated**

**Precision – Recall**

* Với những thông tin có được từ Confusion matrix → có thể định lượng độ hiệu quả của mô hình qua nhiều thang đo khác nhau. Precision và Recall là hai thang đo quan trọng trong số đó.
* **Precision** trả lời cho câu hỏi: trong số các điểm dữ liệu được mô hình phân loại vào lớp Positive, có bao nhiêu điểm dữ liệu thực sự thuộc về lớp Positive.
* **Recall** giúp chúng ta biết được có bao nhiêu điểm dữ liệu thực sự ở lớp Positive được mô hình phân lớp đúng trong mọi điểm dữ liệu thực sự ở lớp Positive.
* Precision và Recall **có giá trị trong [0,1],** hai giá trị này càng gần với 1 thì mô hình càng chính xác.
* Precision **càng cao** đồng nghĩa với các điểm được phân loại **càng chính xác**.
* Recall **càng cao** cho thể hiện cho việc **ít bỏ sót các điểm dữ liệu đúng.**

**A black and white math equation

Description automatically generated**

**A black text with a white background

Description automatically generated**

**A diagram of a positive and negative class

Description automatically generated**

**A white paper with black text and numbers

Description automatically generated**

* Precision và Recall của mô hình này còn thấp → độ chính xác của mô hình chưa cao
* Trong số 30 trường hợp được chẩn đoán dương tính, chỉ có 13 trường hợp thực sự nhiễm COVID-19 (khoảng 43%, được thể hiện qua precision)
* Với giá trị recall = 0.39, ta có thể hiểu rằng trong số 33 trường hợp thật sự dương tính với COVID-19, mô hình chỉ phát hiện ra 13 trường hợp (khoảng 39%).

**F1-Score**

* Một mô hình tốt khi cả Precision và Recall đều cao, thể hiện cho mô hình ít phân loại nhầm giữa các lớp cũng như tỉ lệ bỏ sót các đối tượng thuộc lớp cần quan tâm là thấp
* Tuy nhiên, hai giá trị Precision và Recall thường không cân bằng với nhau (giá trị này tăng thì giá trị kia thường có xu hướng giảm)
* Để đánh giá cùng lúc cả Precision và Recall, ta sử dụng độ đo F-Score

**A screenshot of a computer

Description automatically generated**

* Việc quyết định nên ưu tiên Precision hay Recall phụ thuộc vào từng bài toán.
* Ví dụ, với bài toán xác định một khu vực có bom mìn hay không, việc bỏ sót bom mìn cho hậu quả nghiêm trọng hơn so với khi báo động một khu vực an toàn là có bom, vì vậy cần ưu tiên Recall hơn Precision.
* Ví dụ, việc bỏ sót spam mail có vẻ không tệ nếu so sánh với khi phân loại nhầm một email quan trọng thành spam mail, do đó ở bài toán này, Precision nên được cân nhắc ưu tiên.

A math equation with numbers and symbols

Description automatically generated with medium confidence

* Với những bài toán mà Precision và Recall được cân nhắc ngang nhau, ta chọn β = 1, khi đó ta đang sử dụng F1-Score. F1-Score là kỳ vọng harmonic (harmonic mean) của Precision và Recall.
* F1-score lớn khi cả 2 giá trị Precision và Recall đều lớn. Ngược lại, chỉ cần 1 giá trị nhỏ sẽ làm cho F1-Score nhỏ.

**ROC curve**

* Trong các bài toán phân lớp, các thuật toán phân lớp thường dự đoán điểm số hay xác suất thuộc về một lớp của dữ liệu đầu vào. Điều này giúp ta biết được mức độ chắc chắn của mô hình khi phân lớp.
* Sau khi dự đoán xác suất hay điểm số, cần phải chuyển các giá trị đó về các nhãn của các lớp. Việc chuyển từ xác suất, điểm số sang các nhãn được quyết định bởi một “ngưỡng” (threshold).
* **ROC curve** là một công cụ để chọn ra threshold phù hợp cho mô hình.
* Với mỗi giá trị threshold, ta thu được hai giá trị được biểu diễn trên ROC curve:
  + True Positive Rate (hay Sensitivity – Recall): là độ nhạy của mô hình, cho biết mức độ dự đoán chính xác trong lớp positive. TPR là thương của số điểm dữ liệu được dự đoán đúng thuộc lớp positive với số điểm dữ liệu thuộc lớp positive.

A black text with a white background

Description automatically generated

* Với mỗi giá trị threshold, ta thu được hai giá trị được biểu diễn trên ROC curve:
  + **A black and white text

    Description automatically generated** False Positive Rate: là xác suất mắc Type II Error
* Trong đó, Specificity cho biết mức độ dự đoán chính xác trong lớp negative.

A black text with a white background

Description automatically generated

**A graph of a curve

Description automatically generated with medium confidence**

* True Positive Rate (hay Sensitivity – Recall): là độ nhạy của mô hình
* False Positive Rate: là xác suất mắc Type II Error
* Các điểm màu cam đại diện cho mỗi threshold
* Nối các điểm màu cam lại với nhau ta được ROC curve
* Đường đứt đoạn màu xanh đại diện cho kết quả của “no skill model” – mô hình dự đoán bằng cách random kết quả
* Giá trị FPR càng thấp thì xác suất mắc Type II Error thấp → các điểm ở bên trái nên được cân nhắc hơn nếu ta cần giảm thiểu False Negative (Type II Error).
* Mặt khác, các điểm nằm càng cao thì có TPR càng lớn, đồng nghĩa với việc mô hình có giá trị Recall lớn.
* Tùy thuộc vào bài toán để lựa chọn một điểm – ứng với một threshold phù hợp.

**Area Under the ROC curve (AUC)**

* AUC là phần diện tích bên dưới ROC curve, dùng để đánh giá hiệu năng phân loại của các mô hình với nhau.
* Mô hình có AUC càng lớn (ROC curve càng gần góc cao bên trái) thì cho kết quả càng chính xác.
* Ngược lại, mô hình có ROC curve càng tiến tới đường chéo 45 độ (đường gạch đứt màu xanh ở hình trên), tức AUC càng thấp thì kết quả càng tệ.
* AUC càng cao thì mô hình càng dễ phân loại đúng cho cả lớp positive và negative.

**Underfitting và Overfitting**

* Outlier: Điểm ngoại lai/dị biệt/nhiễu (outlier/noise) là những điểm dữ liệu có giá trị khác thường so với các điểm dữ liệu khác.
* Nhiễu trong dữ liệu huấn luyện khiến cho mô hình “học” sai, dẫn đến một số hậu quả như thời gian huấn luyện sẽ dài hơn, giảm độ chính xác của mô hình và cho kết quả không tốt.
* Nhiễu có thể là kết quả của những sai sót trong quá trình thu thập dữ liệu. Có thể chia nhiễu thành 2 loại cơ bản:
* **Global Outlier:** là điểm dữ liệu **khác hoàn toàn** với toàn bộ các điểm dữ liệu khác trong dataset.
  + Ví dụ: Một vài điểm dữ liệu có giá trị thuộc tính tuổi người lớn hơn 200 thì những điểm dữ liệu đó được xem là global outliers.
* **Contextual Outlier**: là điểm dữ liệu **khác biệt khi so với một phần dữ liệu** trong dataset, nhưng là bình thường khi so với phần dữ liệu khác.
  + Ví dụ: Doanh số bán hàng của một thương hiệu trong các đợt khuyến mãi có thể tăng đột biến so với ngày thường, nhưng không có sự chênh lệch lớn khi so giữa các đợt khuyến mãi với nhau.
* Bias: là sai số giữa giá trị dự đoán trung bình của mô hình và giá trị thực tế.
* High bias: sai số lớn, mô hình đơn giản, tuy nhiên kết quả dự đoán chính xác không cao
* Low bias: sai số nhỏ, mô hình phức tạp, kết quả dự đoán tốt
* Variance: là sai số thể hiện mức độ “nhạy cảm” của mô hình với những biến động trong dữ liệu huấn luyện.
  + - Low-variance: mô hình ít biến thiên theo sự thay đổi của dữ liệu huấn luyện
    - High-variance: mô hình biến thiên mạnh, bám sát theo sự thay đổi của dữ liệu huấn luyện
* Mô hình có variance cao thường thể hiện rất tốt trên tập dữ liệu huấn luyện, nhưng không cho kết quả khả quan trên tập dữ liệu kiểm thử
* **Underfitting**: là hiện tượng mà mô hình có **high bias và low varriance**, cho kết quả dự đoán không tốt trên cả tập huấn luyện và tập kiểm thử.
* Underfitting thường dễ được phát hiện vì cho kết quả tệ trên tập huấn luyện.
* **Overfitting**: là hiện tượng mà mô hình có **low bias và high variance**, lúc này mô hình trở nên phức tạp, bám sát theo dữ liệu huấn luyện.
  + Mô hình cho kết quả rất tốt trên dữ liệu đã được học, nhưng cho kết quả tệ trên dữ liệu chưa từng gặp bao giờ.
  + Vấn đề này xảy ra khi mô hình cố gắng fit tất cả các điểm dữ liệu huấn luyện, bao gồm cả nhiễu.

-> Hiện tượng Underfitting có thể xem như mô hình “học dốt”, còn Overfitting cho biết rằng mô hình đang “học vẹt”

**Nhận biết Underfitting và Overfitting**

* Underfitting xảy ra khi sai số dự đoán của mô hình trên cả tập huấn luyện và tập kiểm thử đều cao.
* Overfitting xảy ra khi sai số dự đoán của mô hình trên tập huấn luyện thấp, nhưng trên tập kiểm thử thì cao
* Cross-validation là một kỹ thuật tốt để ngăn ngừa Overfitting

**Feature engineering - Trích đặc trưng**

* Mỗi điểm dữ liệu trong các bài toán ML thường được biểu diễn bằng một vector được gọi là **vector đặc trưng (feature vector).** Trong cùng một bài toán, các feature vector của tất cả các điểm thường có kích thước như nhau.
* Tuy nhiên, trên thực tế, dữ liệu thường ở dạng thô (raw data) với kích thước khác nhau.
  + Các bài toán thị giác máy tính (computer vision), các bức ảnh thường là các ma trận hoặc tensor với kích thước khác nhau.
  + Các bài toán xử lý ngôn ngữ tự nhiên (natural language processing–NLP ) cũng có khó khăn tương tự khi độ dài của các văn bản là khác nhau, thậm chí có những từ rất hiếm gặp hoặc không có trong từ điển
* Trong ML, việc lựa chọn, tính toán đặc trưng nào phù hợp cho mỗi bài toán là nhiệm vụ quan trọng trước tiên cần được giải quyết.
* Feature extraction **tạo ra một vector đặc trưng** (có kích thước như nhau) cho mỗi điểm dữ liệu đầu vào
* Đầu vào của khối Feature Extraction
  + Dữ liệu thô ban đầu (raw training input)
  + Output của training set
  + Prior knowledge about data: các thông tin khác đã biết về loại dữ liệu (ngoài những thông tin về raw input và output)

**Lưu ý:**

* Dữ liệu thô ban đầu (raw training input): gồm tất cả các thông tin đã biết về dữ liệu
  + VD: Dữ liệu thô của ảnh là giá trị từng pixel; của văn bản là từng từ, từng câu; của file âm thanh là 1 đoạn tín hiệu; ...
* Output của training set: bài toán unsupervised learning không biết output nên sẽ không có giá trị này, bài toán supervised learning, có khi dữ liệu này cũng không được sử dụng
* Sau khi các tham số mô hình của bộ Feature extraction được thiết kế, dữ liệu thô ban đầu được đưa qua và tạo các vector đặc trưng tương ứng → **Extract feature**
* Extract feature được đưa vào huấn luyện các thuật toán (Classification, Regression, Clustering, ...)

**Lưu ý:**

* Trong một số thuật toán cao cấp hơn, việc xây dựng bộ trích chọn đặc trưng và các thuật toán chính (classification, clustering, v.v.) có thể được thực hiện cùng lúc với nhau thay vì từng bước như trên. Các mô hình đó có tên gọi chung là end-to-end.
* Với sự phát triển của deep learning trong những năm gần đây, người ta cho rằng các hệ thống end-to-end (từ đầu đến cuối) mang lại kết quả tốt hơn nhờ vào việc hai khối phía trên được huấn luyện cùng nhau, bổ trợ lẫn nhau cùng hướng tới mục đích cuối cùng.
* Thực tế cho thấy, các phương pháp state-of-the-art (các phương pháp hiệu quả nhất) thường là các mô hình end-to-end.
* Testing phase
  + Khi có dữ liệu thô mới, ta sử dụng bộ trích chọn đặc trưng đã tìm được ở trên để tạo ra vector đặc trưng ứng với dữ liệu thô đó.
  + Vector đặc trưng này được đưa vào thuật toán chính đã tìm được để đưa ra quyết định.

**Một số ví dụ về Feature engineering**

* Trực tiếp lấy dữ liệu thô
* Lựa chọn đặc trưng
* Giảm chiều dữ liệu
* Bag of word (BoW)
* Bag of word trong thị giác máy tính
* Transfer Learning cho bài toán phân loại ảnh
* Chuẩn hóa vector đặc trưng

**Trực tiếp lấy dữ liệu thô**

* Xét một bài toán phân loại các bức ảnh xám mà mỗi bức ảnh đã có kích thước cố định là m × n pixel.
* Cách đơn giản nhất để tạo ra vector đặc trưng cho bức ảnh này là kéo dài ma trận các pixel thành một vector có mn phần tử, hay đặc trưng.
* Khi đó, giá trị mỗi đặc trưng sẽ là một giá trị của một pixel trong bức ảnh ban đầu, thứ tự không quan trọng.
* Kỹ thuật này còn được gọi là vector hoá (vectorization).
* Hạn chế: làm mất thông tin về không gian (spatial information) giữa các điểm ảnh (các pixel gần nhau theo phương ngang trong bức ảnh ban đầu có thể không còn gần nhau trong vector đặc trưng mới nữa)
* Tuy nhiên, trong nhiều trường hợp, kỹ thuật này vẫn mang lại kết quả khả quan.

**Lựa chọn đặc trưng**

* Trường hợp các điểm dữ liệu có **số đặc trưng khác nhau** + **số lượng đặc trưng là cực lớn**
  + Do kích thước dữ liệu khác nhau
  + Do một số đặc trưng mà điểm dữ liệu này có nhưng điểm dữ liệu kia lại không thu thập được
  + → cần chọn ra một số lượng nhỏ hơn các đặc trưng **phù hợp** với bài toán.

**Giảm chiều dữ liệu (Dimensionality Reduction)**

* Nhằm giảm bộ nhớ và khối lượng tính toán
* Không nhất thiết phải giảm số chiều dữ liệu, đôi khi vector đặc trưng có thể có kích thước lớn hơn dữ liệu thô ban đầu

**Bag of words**

* Dữ liệu dạng văn bản (từ, câu, đoạn văn ở dạng text) → vector đặc trưng?

→ Bag of words – BoW: kỹ thuật phổ biến trong xử lý văn bản

* Ví dụ: phân loại tin nhắn rác
  + Tin chứa các từ: khuyến mại, giảm giá, trúng thưởng, miễn phí, quà tặng, tri ân, … → khả năng cao là tin nhắn rác
  + Đếm trong tin nhắn có bao nhiêu từ thuộc các từ trên. Trường hợp số lượng lớn hơn 1 ngưỡng nào đó → tin nhắn rác
  + Với lượng văn bản khác nhau, lượng từ liên quan đến từng chủ đề khác nhau → dựa vào số lượng các từ trong từng loại để tạo các vector đặc trưng cho từng văn bản
* VD: tạo vector đặc trưng từ 2 đoạn văn bản

1. “John likes to watch movies. Mary likes movies too."

2. "John also likes to watch football games."

* + Danh sách các từ được sử dụng → **từ điển (dictionary hoặc codebook)**

["John", "likes", "to", "watch", "movies", "also", "football", "games", "Mary", "too"]

* + Hai vector đặc trưng tương ứng 2 đoạn văn bản

(1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]

(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

* **Lưu ý**:
  + Thực tế, từ điển có nhiều hơn mười từ → vector đặc trưng thu được rất dài
  + Có nhiều từ trong từ điển không xuất hiện trong 1 văn bản → vector đặc trưng thu được thường **có rất nhiều phần tử bằng 0 (vector thưa - sparse vector)** 
    - Để lưu trữ hiệu quả → không lưu cả vector, chỉ lưu **vị trí** các phần tử khác 0 và **giá trị** tương ứng
    - Nếu hơn phân nửa số phần tử khác 0 → phản tác dụng
  + Xử lý các từ hiếm gặp không nằm trong từ điển → thêm phần tử **<Unknow>** vào từ điển → kích thước vector đặc trưng tăng lên 1
  + Đôi khi từ hiếm mang lại thông tin quan trọng → đây là 1 hạn chế của BoW → khắc phục: sử dụng phương pháp **TF-IDF** (Term Frequency - Inverse Document Frequency) để **xác định tầm quan trọng của 1 từ**
  + Hạn chế lớn: *không mang thông tin về thứ tự các từ, sự liên kết giữa các câu, các đoạn trong văn bản*
    - VD: “Em yêu anh không?”, “Em không yêu anh”, “Không, (nhưng) anh yêu em”

Trích chọn bằng Bow cho ra 3 vector giống hệt nhau, mặc dù ý nghĩa khác nhau

**Bag of words trong thị giác máy tính (Computer Vision)**

* BoW áp dụng cho ảnh với cách định nghĩa *từ* và *từ điển* khác
* VD1: có 2 lớp (class) ảnh: khu rừng và sa mạc.
  + Giả sử 1 bức ảnh chỉ thuộc 1 trong 2 lớp, việc phân loại ảnh là dựa vào màu sắc.
  + Màu xanh lục nhiều là rừng, đỏ và vàng nhiều là sa mạc

→ Mô hình đơn giản để trích chọn đặc trưng?

* Mô hình đơn giản để trích chọn đặc trưng
  + Với 1 bức ảnh, chuẩn bị 1 vector x có số chiều là 3: đại diện cho ba màu xanh lục (x1), đỏ (x2), vàng (x3)
  + Với mỗi điểm ảnh trong bức ảnh, ***dựa trên giá trị pixe****l* → xác định gần với màu xanh, đỏ hay vàng. ***Gần với màu nào → tăng giá trị x lên 1***
  + Sau khi xem xét tất cả các điểm ảnh, dù kích thước ảnh khác nhau → thu ***vector kích thước là 3***, mỗi phần tử thể hiện việc có bao nhiêu pixel trong ảnh có màu tương ứng → *histogram vector*
* Trong thực tế, các bài toán xử lý ảnh không chỉ dựa vào màu sắc (có thể dựa vào đường nét, hình dáng, …) → xem xét giá trị từng điểm ảnh không khả quan (lượng thông tin về đường nét bị mất)
* Giải pháp: xem xét 1 vùng hình chữ nhật nhỏ trong ảnh **(patch)**
* Các patch cần đủ lớn để chứa các bộ phận có thể mô tả vật thể trong ảnh
  + VD: mặt người, các patch cần đủ lớn để chứa các phần của khuôn mặt (mắt, mũi, miệng, …)
  + VD: ảnh ô tô, các patch thu được có thể là bánh xe, khung xe, cửa xe, ...
* **Trong văn bản, 2 từ coi là như nhau nếu được biểu diễn bởi ký tự giống nhau**
* **2 patch được coi là như nhau khi Khoảng cách Euclid giữa 2 vector tạo bởi 2 patch nhỏ**

Giả sử xây dựng “từ điển” với khoảng 1000 “từ” với các patch thu được

* Dùng K-means clustering để phân toàn bộ các patch thành 1000 nhóm (bag) khác nhau
* Mỗi nhóm gồm các patch gần giống nhau + được mô tả bằng trung bình cộng của tất cả các patch trong nhóm đó
* Với ảnh bất kỳ, trích các patch từ ảnh đó → tìm nhóm gần mỗi patch nhất (trong 1000 nhóm) → quyết định patch nào thuộc nhóm nào
* Thu được vector đặc trưng có kích thước là 1000, mỗi phần từ là số lượng các patch trong ảnh rơi vào nhóm tương ứng **Transfer learning cho bài toán phân loại ảnh**
  + Lý thuyết về transfer learning được Lorien Pratt thực nghiệm lần đầu năm 1993 và sau đó viết lại nó dưới dạng một lý thuyết toán học vào năm 1998 đã hiện thực hóa ý tưởng về chuyển giao tri thức giữa các mô hình như giữa con người với nhau.
  + Một mô hình có khả năng tận dụng lại các tri thức đã huấn luyện trước đó và cải thiện lại trên tác vụ phân loại của nó
    - Cải thiện accuracy và tiết kiệm chi phí huấn luyện
    - Hiệu quả với dữ liệu nhỏ
  + **Cải thiện accuracy và tiết kiệm chi phí huấn luyện**

Từ đồ thị ta có thể thấy sử dụng transfer learning sẽ mang lại 3 lợi thế chính:

- Có điểm khởi đầu của accuracy tốt hơn (higher start).

- Accuracy có tốc độ tăng nhanh hơn (higher slope).

- Đường tiệm cận của độ chính xác tối ưu cao hơn (higher asymptote).

A diagram of a training curve

Description automatically generated

* **Hiệu quả với dữ liệu nhỏ**
  + Trong trường hợp bộ dữ liệu có kích thước quá nhỏ và khó có thể tìm kiếm và mở rộng thêm thì các mô hình được huấn luyện từ chúng sẽ khó có thể dự báo tốt.
  + Tận dụng lại tri thức từ các pretrained-model với cùng tác vụ phân loại sẽ giúp các mô hình được huấn luyện dự báo tốt hơn với dữ liệu mới vì mô hình được học trên cả 2 nguồn tri thức đó là dữ liệu huấn luyện và dữ liệu mà nó đã được học trước đó.
  + Ví dụ: Trong bài toán dự báo dog and cat
    - Có 2 nhãn cần phân loại là dog, cat
    - Cả 2 nhãn này đều xuất hiện trong một bộ dữ liệu imagenet

→ kì vọng rằng có thể tận dụng lại các weights từ pretrained-model trên bộ dữ liệu imagenet để huấn luyện lại bài toán nhanh hơn, chuẩn xác hơn.

* Dựa trên kích thước và độ tương quan giữa CSDL mới và CSDL gốc (chủ yếu là ImageNet), có một vài quy tắc để huấn luyện network mới như sau
  + ***CSDL mới là nhỏ và tương tự như CSDL gốc***
    - CSDL mới nhỏ, việc tiếp tục huấn luyện mô hình có thể dễ dẫn đến hiện tượng overfitting
    - Các high-level feature tương tự nhau
    - không cần huấn luyện lại network mà chỉ cần huấn luyện một classifer dựa trên feature vector ở đầu ra ở layer gần cuối
  + ***CSDL mới là lớn và tương tự như CSDL gốc***
    - CSDL này lớn, overfitting ít có khả năng xảy ra hơn, ta có thể huấn luyện mô hình thêm một một vài vòng lặp
    - Việc huấn luyện có thể được thực hiện trên toàn bộ hoặc chỉ một vài layer cuối.
* Dựa trên kích thước và độ tương quan giữa CSDL mới và CSDL gốc (chủ yếu là ImageNet), có một vài quy tắc để huấn luyện network mới như sau
  + ***CSDL mới là nhỏ và rất khác với CSDL gốc***
    - CSDL này nhỏ, tốt hơn hết là dùng các classifier đơn giản để tránh overfitting
    - Nếu muốn huấn luyện thêm, chỉ nên thực hiện trên các layer cuối
    - Hoặc sử dụng một kỹ thuật khác là coi đầu ra của một layer xa layer cuối hơn (xa hơn layer gần cuối) làm các feature vector
* Dựa trên kích thước và độ tương quan giữa CSDL mới và CSDL gốc (chủ yếu là ImageNet), có một vài quy tắc để huấn luyện network mới như sau
  + ***CSDL mới là lớn và rất khác CSDL gốc***
    - Thực tế cho thấy, sử dụng các network sẵn có trên CSDL mới vẫn hữu ích
    - Trong trường hợp này, ta vẫn có thể sử dụng các network sẵn có như là điểm khởi tạo của network mới, không nên huấn luyện network mới từ đầu.
* Khi tiếp tục huấn luyện các network này, ta chỉ nên chọn learning rate nhỏ để các hệ số mới không đi quá xa so với các hệ số đã được huấn luyện ở các mô hình trước

**Chuẩn hóa vector đặc trưng**

* + Thực hiện khi vector dữ liệu đã có cùng chiều
  + Các phương pháp chuẩn hóa thường dùng
    - Rescaling
    - Standardization
    - Scaling to unit length
  + **Rescaling**
    - Đưa tất cả các đặc trưng về cùng 1 khoảng ([0,1] hoặc [-1,1]), tùy thuộc vào ứng dụng
    - Công thức: đưa đặc trưng thứ i của 1 vector đặc trưng x về khoảng [0,1]
    - Trong đó: xi : giá trị đặc trưng ban đầu
    - x’i : giá trị đặc trưng sau khi được chuẩn hóa
    - min(xi), max(xi) : giá trị nhỏ nhất và lớn nhất của đặc trưng thứ i xét trên toàn bộ các điểm dữ liệu của tập huấn luyện
  + **Standardization**
    - Giả sử mỗi đặc trưng đều có phân phối chuẩn với kỳ vọng là 0, phương sai là 1
    - Công thức chuẩn hóa

A mathematical equation with black text

Description automatically generated

* + - Trong đó: x̄ i , σi: kỳ vọng và độ lệch chuẩn (standard deviation) của đặc trưng trên toàn bộ dữ liệu huấn luyện
  + **Scaling to unit length**
    - Chuẩn hóa các thành phần của mỗi vector dữ liệu sao cho toàn bộ vector có độ dài Euclid bằng 1
    - A black and white math equation

      Description automatically generated Công thức