CÁC BÀI TOÁN CƠ BẢN

Clustering (1)

- Phân cụm (Clustering): Mục tiêu của phân cụm là tìm ra các nhóm dữ liệu có các đặc điểm tương đồng, chẳng hạn như xếp các bức ảnh khuôn mặt gần giống nhau vào các nhóm (nhưng không biết bức ảnh đó của ai và là gì). Đây là khả năng học tập cơ bản nhất của con người - khả năng so sánh sự tương đồng. Dữ liệu được sử dụng trong bài toán phân cụm không được gán nhãn, nhưng có độ tương tự về cấu trúc tự nhiên của dữ liệu.
- Một vấn đề phức tạp nhất của bài toán phân cụm là chúng ta không sao biết được việc gom nhóm các đối tượng có "đúng" với bản chất của nó hay không?, cùng một bộ dữ liệu nhưng có thể có vô số cách phân các nhóm khác nhau cho dữ liệu này. chúng ta không hề biết cách nào ở trên là đúng.
- Sở dĩ xuất hiện vấn đề này là do chúng ta không hề biết trước các nhãn của dữ liệu và do đó, không biết phải phân bao nhiêu nhóm (cụm), và sự tương đồng được đánh giá như thế nào hay dựa trên tiêu chí nào

Clustering (2)

- Để biết được việc gom cụm có đúng hay không thường dựa vào kiến thức chuyên gia, tức là ấn định các nhãn cho cụm dựa trên một đặc tính đã biết trước do chuyên gia chỉ định.
- Ví dụ: Trong lĩnh vực an ninh mạng, chúng ta có thể phân cụm các lưu lượng vào mạng thành các nhóm khác nhau, và chuyên gia an toàn thông tin sẽ phải xác định xem các nhóm này, nhóm nào là lưu lượng độc hại và nhóm nào không, từ đó mới có được đánh giá độ chính xác cho mô hình này. Đây là cách đánh giá gián tiếp cho mô hình phân cụm thông qua gán nhãn lớp. Về mặt mô hình, độ đo P để đánh giá nhiệm vụ phân cụm ở đây là tổng khoảng cách (độ tương đồng) giữa các đối tượng trong nhóm (cụm) sao cho tổng khoảng cách này là nhỏ nhất có thể.
- Do, việc phân nhóm phụ thuộc rất lớn vào các tiêu chí, vì vậy, cách lựa chọn thuộc tính (tiêu chí) để đánh giá là yếu tố quyết định để phân nhóm chính xác. Việc càng có nhiều thuộc tính thừa trong dữ liệu chỉ làm mô hình phân cụm bị sai lệch và không thể sử dụng được.

Classification (1)

- Phân lớp (Classification): Khác với phân cụm, phân lớp dễ dàng đánh giá hơn. Chúng ta cần có sẵn các dữ liệu được gán nhãn sẵn thành các lớp, chẳng hạn như: độc hại/không độc hại, gian lận/không gian lận,...
- Nhiệm vụ của mô hình là quyết định gán nhãn hay phân lớp cho các dữ liệu chưa được gán nhãn. Bài toán phân lớp là một trong những bài toán có ứng dụng rộng rãi nhất trên thực tế, và độ đo đánh giá của nó cũng rất rõ ràng (dựa trên tỷ lệ chính xác, tỷ lệ phân lớp đúng/sai, và một số tiêu chí khác sẽ trình bày trong bài tiếp theo).

Regression

- Hồi quy (Regression):. Việc mô hình hóa và xây dựng các quy tắc tổng quát để mô tả thế giới. Về mặt bản chất, các công thức Newton hay các phương trình nhiệt động,... và tất cả các phương trình được xây dựng trong khoa học đều là bài toán hồi quy.
- Nghĩa là từ một tập các dữ liệu quan sát được, các nhà khoa học sẽ tìm cách để xây dựng các công thức biểu diễn để mô tả và tổng quát hóa tự nhiên. Các công thức này đều được coi là đúng cho đến khi tồn tại các bộ dữ liệu chứng minh nó sai
- Về mặt bản chất hồi quy là bài toán gán nhãn cho dữ liệu thực, biểu diễn và dự đoán đầu ra dựa trên tổng quát hóa các dữ liệu từ đầu vào để tìm ra một hàm dự đoán. Chẳng hạn xây dựng hàm dự đoán giá nhà, giá cổ phiếu theo thời gian hoặc các biến đầu vào khác. Nếu chú ý có thể thấy rằng, thực chất bài toán phân lớp cũng là một trường hợp đặc biệt của bài toán hồi quy, khi giá trị các dự đoán đầu ra thay vì các lớp thì là các giá trị rời rạc đại diện cho lớp.
- Điểm khó khăn nhất với hồi quy là xác định hàm hồi quy, và vấn đề gây rắc rối nhất là các dữ liệu nhiễu hoặc quan sát dữ liệu sai. Điều này sẽ làm cho các công thức hoàn toàn bị sai hoặc gây hiểu nhầm.

Rule Extraction

- Trích rút luật (Rule Extraction): Đây là bài toán mà dữ liệu được sử dụng để tìm ra các quy luật.
- Bài toán này cũng giống như hồi quy, về mặt bản chất đều xuất phát từ vấn đề học cách tổng quát hóa và dự đoán của con người. Tuy nhiên thay vì dự đoán các giá trị thực thì trích rút luật tổng quát hóa các luật, các mẫu suy luận nếu - thì từ dữ liệu luôn đặt ra. Các luật này có thể không trực tiếp thậm chí không thể hiểu được dưới dạng suy luận của con người.
- Việc tìm ra luật dựa trên các nguyên tắc thống kê quan hệ giữa các thuộc tính và dữ liệu mà không liên quan đến các tri thức tiên nghiệm.

Một số bài toán khác

- Ước lượng mật độ (Density estimation): liên quan đến bài toán biểu diễn dữ liệu, tìm kiếm các phân bố của dữ liệu đầu vào trong một số không gian. Bài toán này liên quan đến các vấn đề dự đoán và ước lượng phân bố xác suất của dữ liệu, và có ứng dụng khá lớn trong các bài toán liên quan đến các biến và mô hình ngẫu nhiên như bài toán dự báo tỷ giá, dự đoán thị trường hoặc trong lĩnh vực y tế.
- Giảm chiều (Dimensionality reduction): bài toán này liên quan đến vấn đề chọn lựa các thuộc tính quan trọng phù hợp của dữ liệu bằng cách biến đổi các dữ liệu sang không gian có số chiều nhỏ hơn. Một ứng dụng của giảm chiều là bài toán mô hình hóa chủ đề của văn bản, trong đó một chương trình được cho một danh sách các văn bản ngôn ngữ thông thường và nhiệm vụ là tìm kiếm các văn bản có cùng chủ đề. Và ứng dụng thứ hai của nó là làm giảm độ phức tạp dữ liệu, hay quan trọng hơn là tạo ra các bộ dữ liệu "tốt" hơn theo đúng tiêu chí.
- Học để học (meta-learning): Đây là một bài toán mới trong học máy, một chủ đề mới nổi lên gần đây. Mục tiêu của meta learning là học cách để học, tức là học cách để chọn ra mô hình phù hợp nhất nhằm cải thiện độ chính xác.

Hàm mất mát

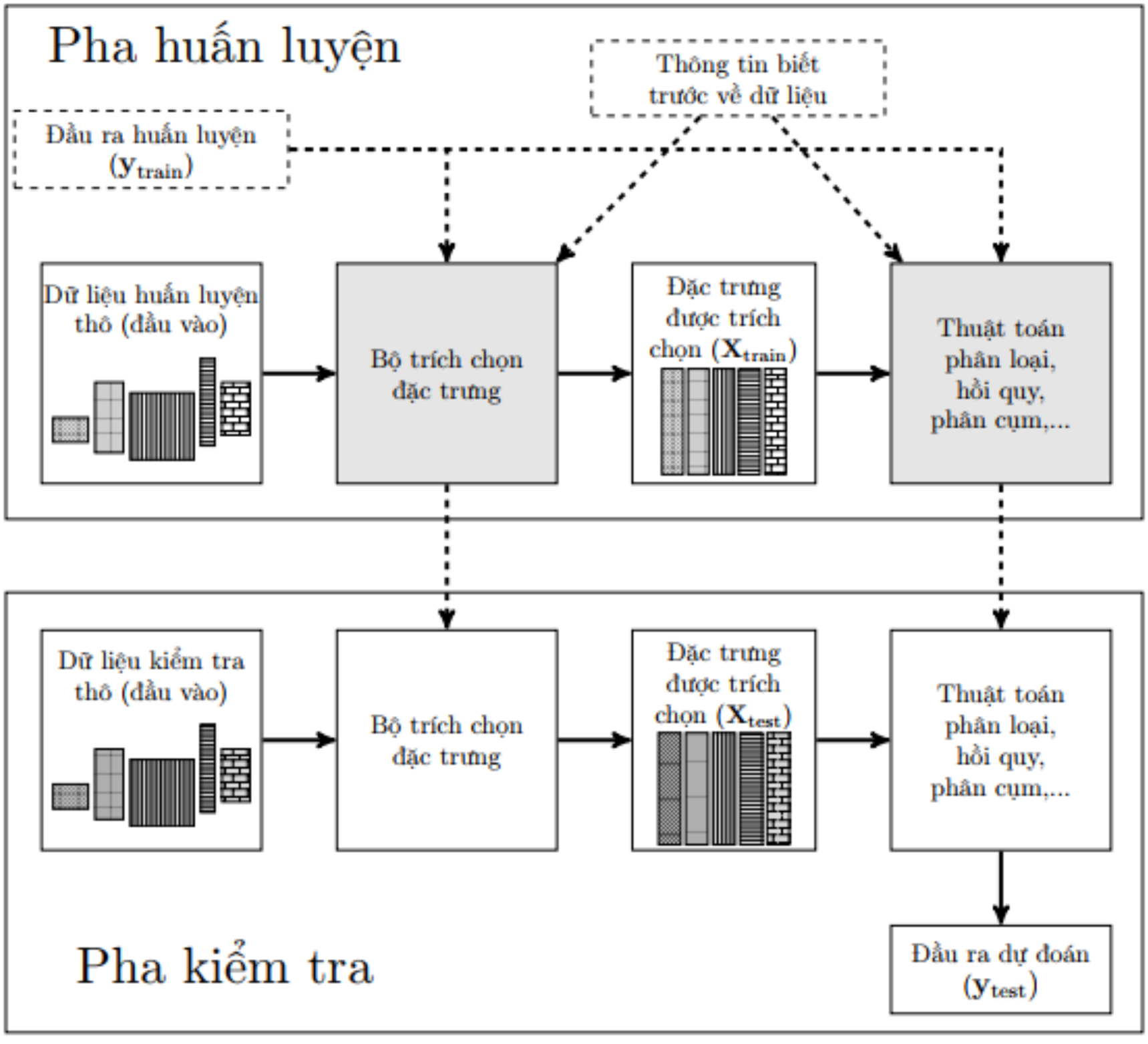
- Định lượng mức độ tốt hoặc tệ như thế nào tròng việc phân loại các điểm dữ liệu đầu vào trong một tập dữ liệu.
- Hàm mất mát càng nhỏ, công việc phân loại càng tốt trong việc mô hình hóa mối quan hệ giữa dữ liệu đầu vào và nhãn lớp đầu ra. Ngược lại, tổn thất của chúng ta càng lớn, càng cần phải làm nhiều việc hơn để tăng phân loại chính xác.

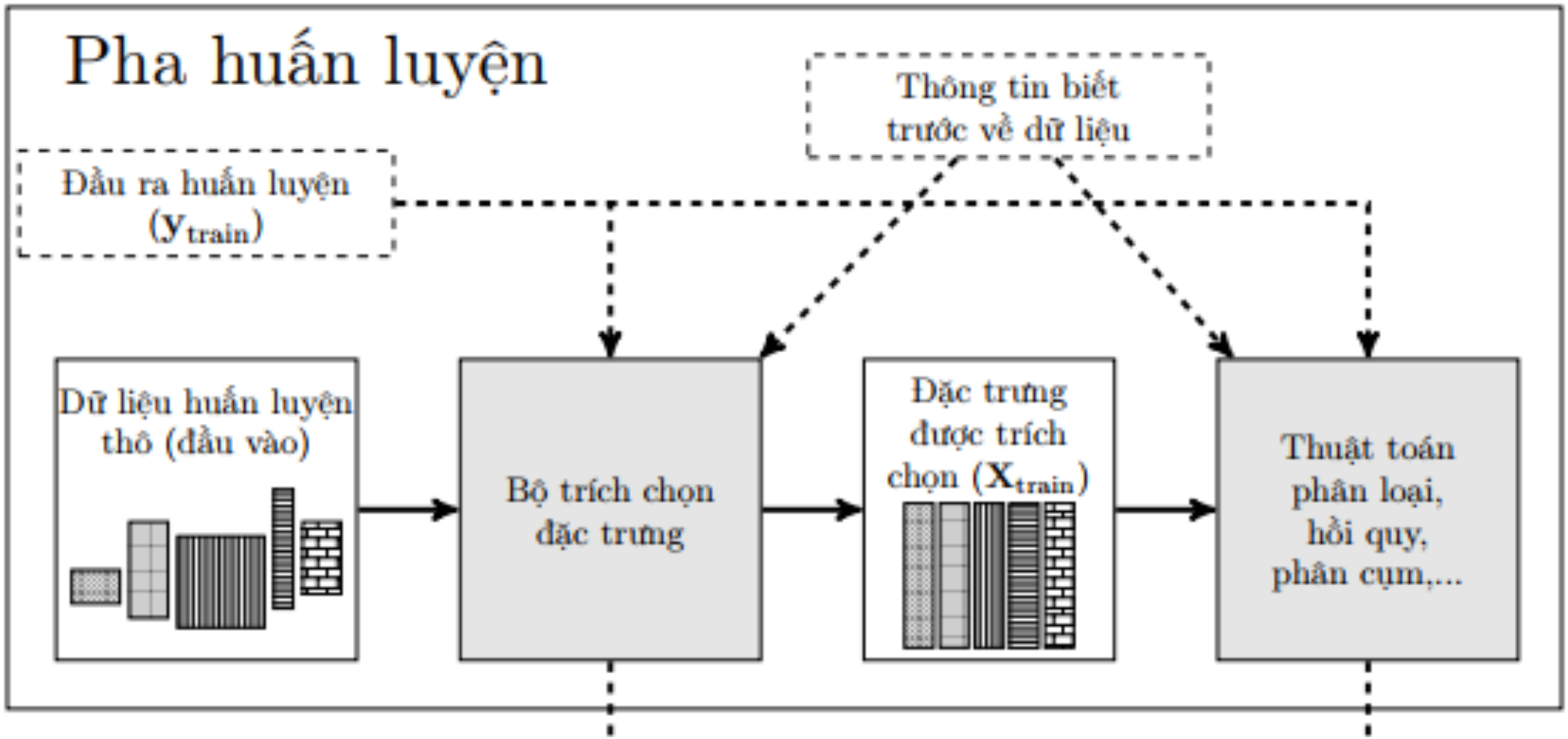
Để cải thiện độ chính xác phân loại cần điều chỉnh các tham số của ma trận trọng số chính xác thì cách chúng ta tiến hành cập nhật các tham số này là tối ưu hóa, hàm mất mát có thể được sử dụng để định lượng chức năng hoạt động tốt như thế nào khi phân loại đầu vào điểm dữ liệu.

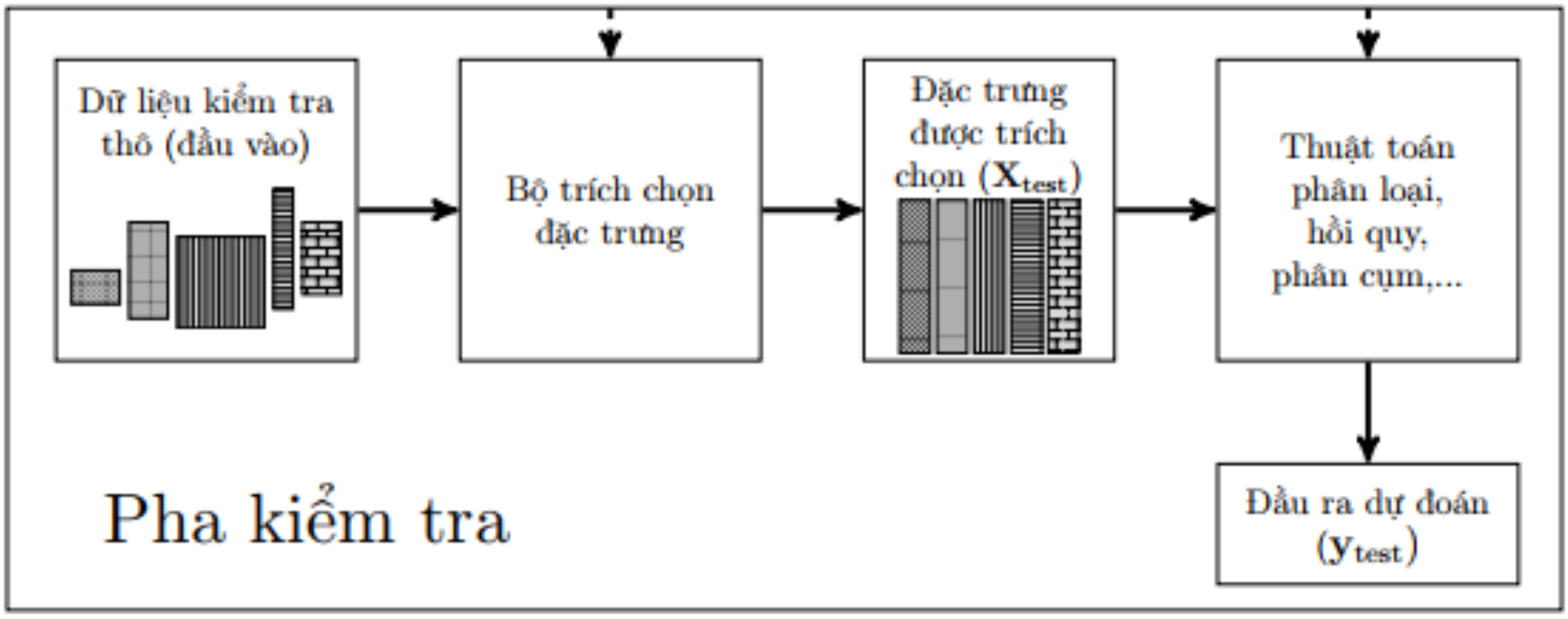
Lý tưởng nhất là tổn thất sẽ giảm theo thời gian khi điều chỉnh các tham số mô hình.

CÁC KỸ THUẬT XÂY DỰNG ĐẶC TRƯNG

Mô hình chung




Pha huấn luyện
Thông tin biết trước về dữ liệu
Đầu ra huấn luyện (y_{train})
Dữ liệu huấn luyện thô (đầu vào)
Bộ trích chọn đặc trưng
Đặc trưng được trích chọn (X_{train})
Thuật toán phân loại, hồi quy, phân cụm,...


Dữ liệu kiểm tra thô (đầu vào)
Bộ trích chọn đặc trưng
Đặc trưng được trích chọn ($\mathbf{X}_{test}$)
Thuật toán phân loại, hồi quy, phân cụm,...
Đầu ra dự đoán ($\mathbf{y}_{test}$)
Pha kiểm tra

Bộ trích chọn đặc trưng
Feature Engineering

- Là một kĩ thuật giúp giảm chiều giữ liệu mà ở đó cho phép chúng ta lựa chọn hoặc kết hợp các biến đầu vào thành những *đặc trưng* dự báo nhưng vẫn thể hiện một cách chính xác và nguyên vẹn của dữ liệu gốc.
- ***Biến đổi đặc trưng*** là những kĩ thuật giúp biến đổi dữ liệu đầu vào thành những dữ liệu phù hợp với mô hình nghiên cứu.
- ***Lựa chọn đặc trưng:*** Rất quan trọng trong Machine Learning với mục tiêu chính là loại bỏ những đặc trưng không thực sự chứa thông tin hữu ích cho bài toán phân loại hoặc dự báo. Sử dụng để cải thiện tốc độ huấn luyện và dự báo và giảm hiện tượng *quá khớp*.
- **Trích lọc đặc trưng:** làm sạch và biến đổi thành dạng dữ liệu cấu trúc từ dữ liệu dạng thô và đến từ nhiều nguồn khác nhau như văn bản, hình ảnh, âm thanh, các phiếu điều tra, các hệ thống lưu trữ, website, app,…

Một số ví dụ về Feature Engineering

- **Trực tiếp lấy raw data**
- Với bài toán phân loại chữ số viết tay trong bộ cơ sở dữ liệu MNIST, mỗi bức ảnh có số chiều là 28 pixel x 28 pixel (tất nhiên việc *crop* và chỉnh sửa mỗi bức ảnh đã được thực hiện từ trước rồi, đó đã là một phần của feature engineering rồi). Một cách đơn giản thường được dùng là *kéo dài* ma trận 28x28 này để được 1 vector có số chiều 784. Trong cách này, các cột (hoặc hàng) của ma trận ảnh được đặt chồng lên (hoặc cạnh nhau) để được 1 vector dài. Vector dài này được trực tiếp sử dụng làm feature đưa vào các bộ classifier/clustering/regression/… Lúc này, giá trị của mỗi pixel ảnh được coi là một feature.
- Rõ ràng việc làm đơn giản này đã làm mất thông tin về *không gian* (spatial information) giữa các điểm ảnh, tuy nhiên, trong nhiều trường hợp, nó vẫn mang lại kết quả khả quan.
- **Lựa chọn tính năng**
- Giả sử rằng các điểm dữ liệu có số features khác nhau (do kích thước dữ liệu khác nhau hay do một số feature mà điểm dữ liệu này có nhưng điểm dữ liệu kia lại không thu thập được), và số lượng features là cực lớn. Chúng ta cần *chọn* ra một số lượng nhỏ hơn các feature phù hợp với bài toán. *Chọn thế nào* và *thế nào là phù hợp* lại là một bài toán khác, tôi sẽ không bàn thêm ở đây.

Một số ví dụ về Feature Engineering

- **Giảm kích thước**
- Một phương pháp nữa tôi đã đề cập đó là làm giảm số chiều của dữ liệu để giảm bộ nhớ và khối lượng tính toán. Việc giảm số chiều này có thể được thực hiện bằng nhiều cách, trong đó *random projection* là cách đơn giản nhất. Tức chọn một *ma trận chiếu* (projection matrix) ngẫu nhiên (ma trận béo) rồi nhân nó với từng điểm dữ liệu (giả sử dữ liệu ở dạng vector cột) để được các vector có số chiều thấp hơn. Ví dụ, vector ban đầu có số chiều là 784, chọn *ma trận chiếu* có kích thước (100x784), khi đó nếu nhân ma trận chéo này với vector ban đầu, ta sẽ được một vector mới có số chiều là 100, nhỏ hơn số chiều ban đầu rất nhiều. Lúc này, có thể ta không có tên gọi cho mỗi feature nữa vì các feature ở vector ban đầu đã được trộn lẫn với nhau theo một tỉ lệ nào đó rồi lưu vào vector mới này. Mỗi thành phần của vector mới này được coi là một feature (không tên).
- Việc chọn một ma trận chiếu ngẫu nhiên đôi khi mang lại kết quả tệ không mong muốn vì thông tin bị mất đi quá nhiều. Một phương pháp được sử dụng nhiều để hạn chế lượng thông tin mất đi có tên là Principle Component Analysis sẽ được tôi trình bày sau đây khoảng 1-2 tháng.
- **Chú ý:** Feature learning không nhất thiết phải làm giảm số chiều dữ liệu, đôi khi feature vector còn có số chiều lớn hơn raw data. Random projection cũng có thể làm được việc này nếu ma trận chiếu là một ma trận *cao* (số cột ít hơn số hàng).

Một số ví dụ về Feature Engineering

- **Túi từ (***Bag of Words (BoW)*
- Hẳn rất nhiều bạn đã tự đặt câu hỏi: Với một văn bản thì feature vector sẽ có dạng như thế nào? Làm sao đưa các từ, các câu, đoạn văn ở dạng *text* trong các văn bản về một vector mà mỗi phần tử là một số?
- Có một phương pháp rất phổ biến giúp bạn trả lời những câu hỏi này. Phương pháp đó có tên là *Bag of Words (BoW)* (*Túi đựng Từ*).
- Vẫn theo thói quen, tôi bắt đầu bằng một ví dụ. Giả sử chúng ta có bài toán phân loại tin rác. Ta thấy rằng nếu một tin có chứa các từ *khuyến mại, giảm giá, trúng thưởng, miễn phí, quà tặng, tri ân, …* thì nhiều khả năng đó là một tin nhắn rác. Vậy phương pháp đơn giản nhất là *đếm* xem trong tin đó có bao nhiêu từ thuộc vào các từ trên, nếu nhiều hơn 1 ngưỡng nào đó thì ta quyết định đó là tin rác. (Tất nhiên bài toán thực tế phức tạp hơn nhiều khi các từ có thể được viết dưới dạng không dấu, hoặc bị cố tình viết sai chính tả, hoặc dùng ngôn ngữ teen). Với các loại văn bản khác nhau thì lượng từ liên quan tới từng chủ đề cũng khác nhau. Từ đó có thể dựa vào số lượng các từ trong từng loại để làm các vector đặc trưng cho từng văn bản.

Một số ví dụ về Feature Engineering

Giả sử chúng ta có hai văn bản đơn giản:

```
(1) John likes to watch movies. Mary likes movies too.
```

và

```
(2) John also likes to watch football games.
```

Dựa trên hai văn bản này, ta có danh sách các từ được sử dụng, được gọi là *từ điển* với 10 *từ* như sau:

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

Với mỗi văn bản, ta sẽ tạo ra một vector đặc trưng có số chiều bằng 10, mỗi phần tử đại diện cho số từ tương ứng xuất hiện trong văn bản đó. Với hai văn bản trên, ta sẽ có hai vector đặc trưng là:

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

Văn bản (1) có 1 từ “John”, 2 từ “likes”, 0 từ “also”, 0 từ “football”, … nên ta thu được vector tương ứng như trên.

Một số ví dụ về Feature Engineering

Túi từ trong Thị giác máy tính

- **Ví dụ 1:**

Có hai class ảnh, một class là ảnh các khu rừng, một class là ảnh các sa mạc. Phân loại một bức ảnh là rừng hay sa mạc (giả sử ta biết rằng nó thuộc một trong hai loại này) một cách trực quan nhất là dựa vào màu sắc. Màu xanh nhiều thì là rừng, màu đỏ và vàng nhiều thì là sa mạc. Vậy chúng ta có thể có một mô hình đơn giản để trích chọn đặc trưng như sau:

Với một bức ảnh, chuẩn bị một vector x

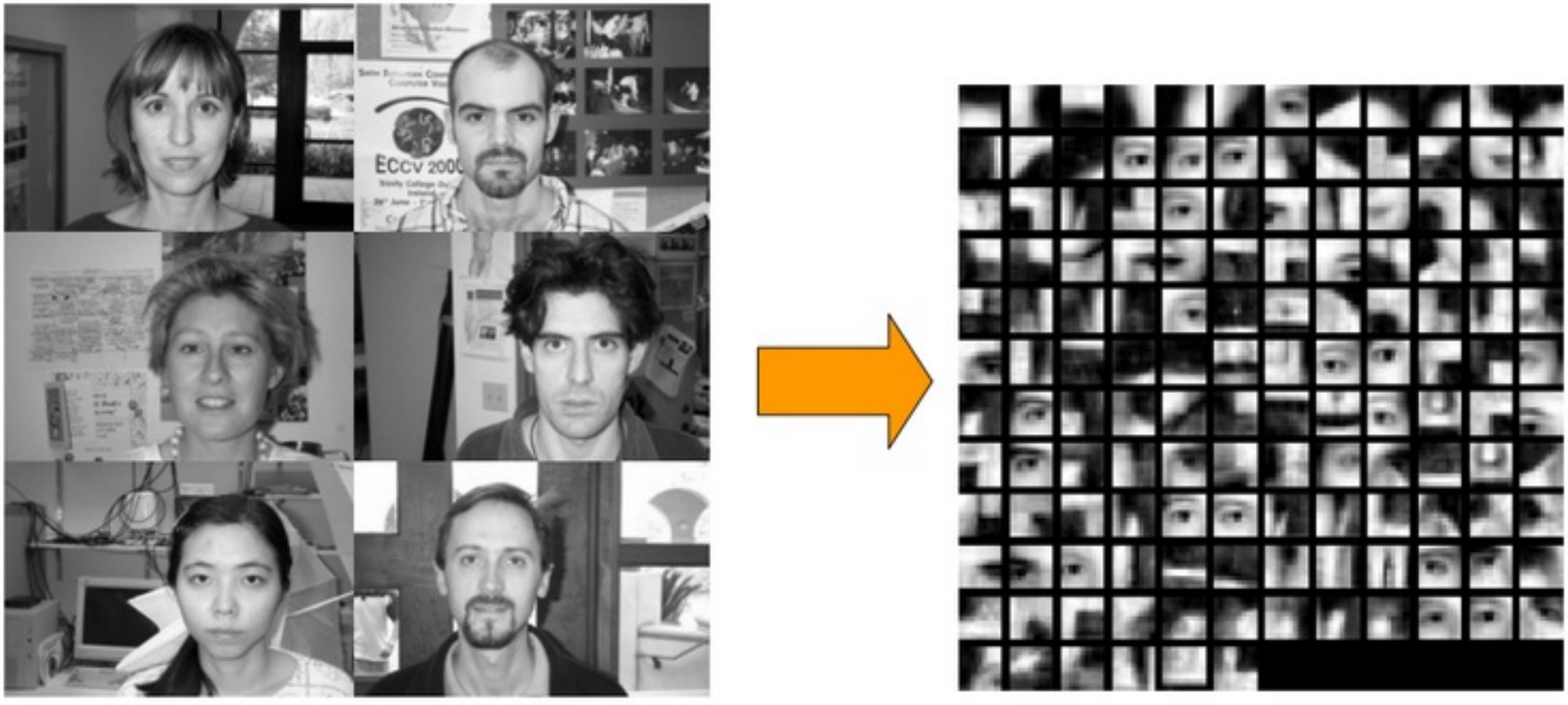
có số chiều bằng 3, đại diện cho 3 màu xanh (x1), đỏ (x2), và vàng (x3).

Với mỗi điểm ảnh trong bức ảnh đó, xem nó gần với màu xanh, đỏ hay vàng nhất dựa trên giá trị của pixel đó. Nếu nó gần điểm xanh nhất, tăng x1

lên 1; gần đỏ nhất, tăng x2 lên 1; gần vàng nhất, tăng x3lên 1.

Sau khi xem xét tất cả các điểm ảnh, dù cho bức ảnh có kích thước thế nào, ta vẫn thu được một vector có độ dài bằng 3, mỗi phần tử thể hiện việc có bao nhiêu pixel trong bức ảnh có màu tương ứng. Vector cuối này còn được gọi là vector histogram của bức ảnh tương ứng với ba màu xanh, đỏ, vàng. Dựa vào vector này, ta có thể quyết định bức ảnh đó là ảnh rừng hay sa mạc.

Một số ví dụ về Feature Engineering



Túi từ trong Thị giác máy tính

Một số ví dụ về Feature Engineering

Chia tỷ lệ và chuẩn hóa tính năng

Các điểm dữ liệu đôi khi được đo đạc với những đơn vị khác nhau, m và feet chẳng hạn. Hoặc có hai thành phần (của vector dữ liệu) chênh lệch nhau quá lớn, một thành phần có khoảng giá trị từ 0 đến 1000, thành phần kia chỉ có khoảng giá trị từ 0 đến 1 chẳng hạn. Lúc này, chúng ta cần chuẩn hóa dữ liệu trước khi thực hiện các bước tiếp theo.

Một vài phương pháp chuẩn hóa thường dùng:

Thay đổi kích thước

Tiêu chuẩn hóa

Một số ví dụ về Feature Engineering

- **Thay đổi kích thước**
- Phương pháp đơn giản nhất là đưa tất cả các thành phần về cùng một khoảng, [0,1] hoặc [−1,1] chẳng hạn, tùy thuộc vào ứng dụng. Nếu muốn đưa một thành phần (feature) về khoảng [0,1], công thức sẽ là:

$$x^{'} = \frac{x - \text{phút}(x)}{\text{tối đa}(x) - \text{phút}(x)}$$

- trong đó x là giá trị ban đầu, x' là giá trị sau khi chuẩn hóa. phút(x),tối đa(x) được tính trên toàn bộ dữ liệu training data ở cùng một thành phần. Việc này được thực hiện trên từng thành phần của vector dữ liệu x.

Một số ví dụ về Feature Engineering

- **Tiêu chuẩn hóa**
- Một phương pháp nữa cũng hay được sử dụng là giả sử mỗi thành phần đều có phân phối chuẩn với kỳ vọng là 0 và phương sai là 1. Khi đó, công thức chuẩn hóa sẽ là:

$$x^{'} = \frac{x - \text{phút}(x)}{\text{tối đa}(x) - \text{phút}(x)}$$

- trong đó x là giá trị ban đầu, x' là giá trị sau khi chuẩn hóa. phút(x),tối đa(x) được tính trên toàn bộ dữ liệu training data ở cùng một thành phần. Việc này được thực hiện trên từng thành phần của vector dữ liệu x.