

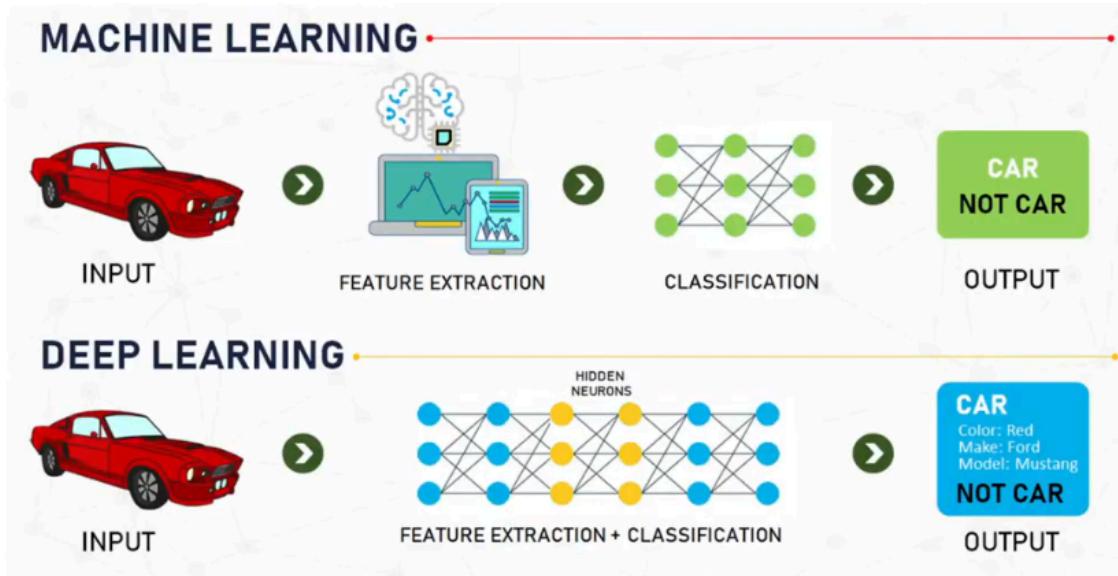


DeepLearning

⇒ DeepLearning

Bài toán Image Classification là helloworld của computer vision

ONNX là định dạng trung gian ⇒ khi convert từ A sang B như pythot sang tensorflow
tensor RT định dạng riêng của NVidia

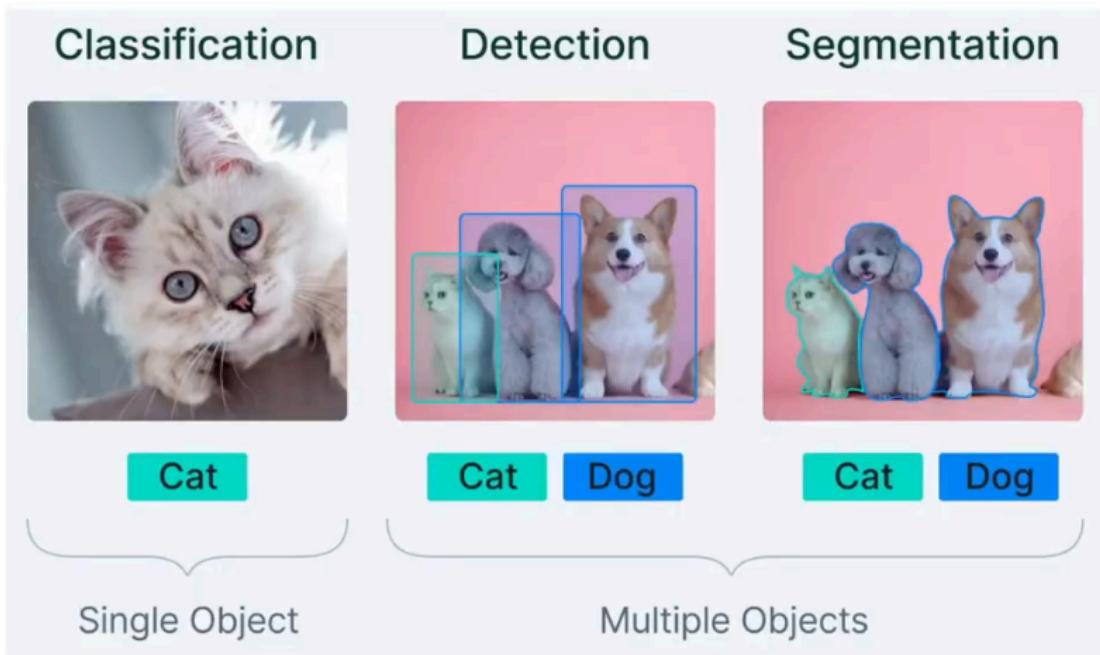


Tầng gần input thì extract những feature đơn giản - nét thẳng...

Tầng gần output thì extract những feature phức tạp hơn

Ba bài toán thường gặp trong computer vision

Deep Learning for Computer Vision



Bản chất ảnh được tạo ra bởi các pixel - có giá trị từ 0 đến 255

nếu ĐT \Rightarrow biểu diễn bằng mảng 2 chiều có kích thước bằng kích thước bức ảnh
(giá trị 0 - đến hoặc 255)

nếu là ảnh màu \Rightarrow dùng mảng 3 chiều

số hàng / số cột và channel màu

Ảnh màu (RGB) hoặc có trong suốt RGBA \rightarrow Mảng 3 chiều

- Mỗi pixel có 3 giá trị (R, G, B), nên ta dùng **mảng 3 chiều**:

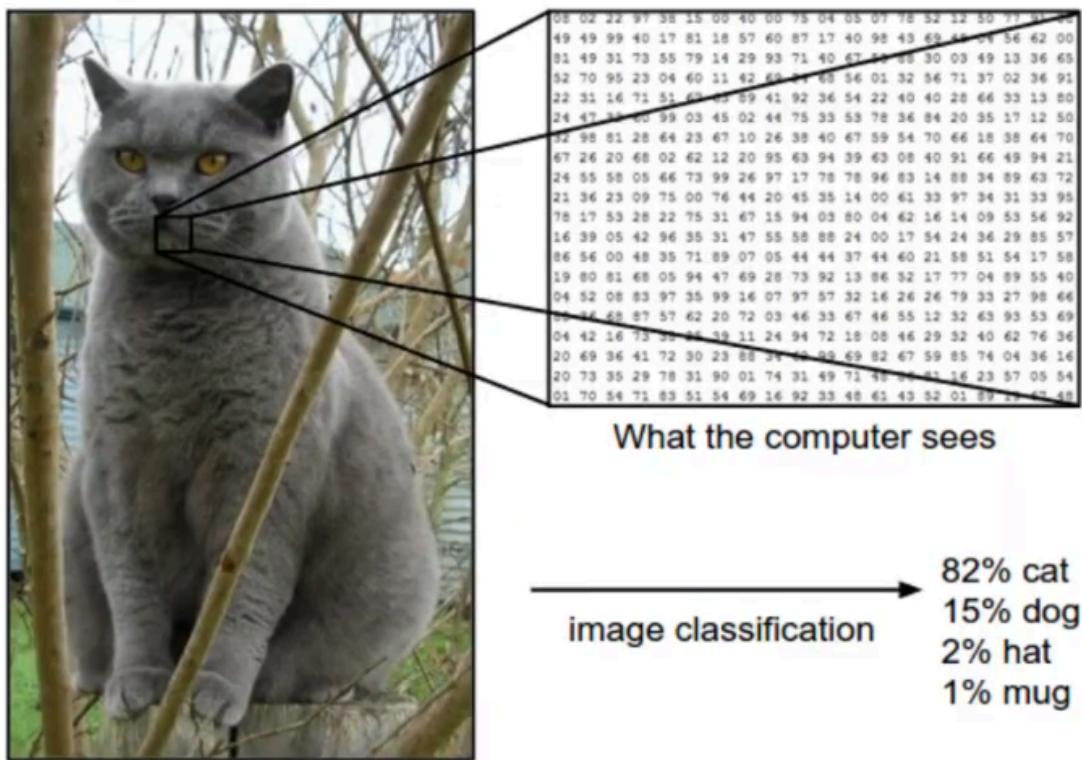
height x width x 3 (R, G, B)

- Ví dụ:

lua

CopyEdit

```
[[[255, 0, 0], [0, 255, 0], [0, 0, 255]], # Dòng 1  
[[255, 255, 0], [0, 255, 255], [255, 0, 255]], # Dòng 2  
[[0, 0, 0], [128, 128, 128], [255, 255, 255]]] # Dòng 3
```

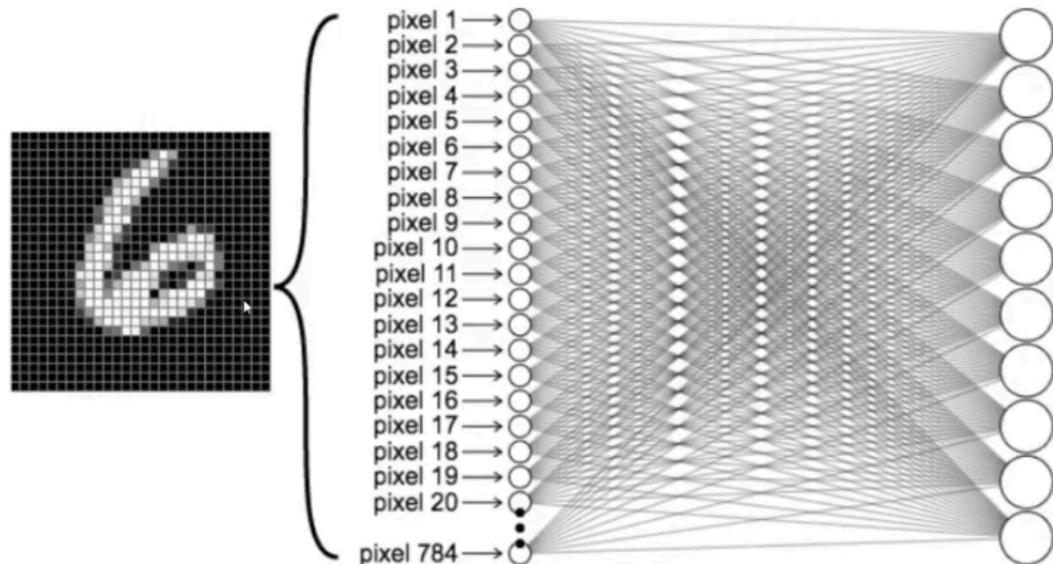
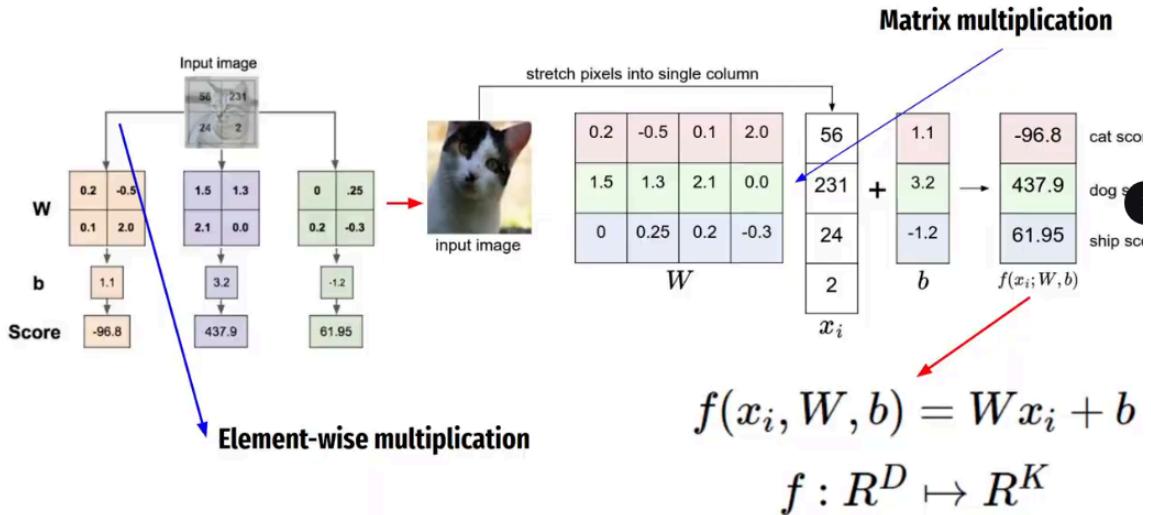


CVAT là công cụ để đánh nhãn dữ liệu

Một mô hình trong deeplearning sẽ có 3 phần chính:

Score function : Tính điểm - làm sao để mapping input (mảng 2 chiều có nhiều phần tử) thành output (mèo, gà...)

Linear Classifier - score function



lost function: hàm mất mát

optimization function: hàm tối ưu

Neuron = linear classifier + active function

activate function sẽ quyết định bao nhiêu lượng dữ liệu sẽ được đi qua + thêm tính chất phi tuyến tính vào mô hình

Hàm kích hoạt (activation function) trong mạng nơ-ron (neural network) có vai trò rất quan trọng. Dưới đây là các tác dụng chính của hàm kích hoạt:

1. Tính phi tuyến tính (Non-linearity):

Hàm kích hoạt giúp mạng nơ-ron có khả năng học và biểu diễn các quan hệ phi tuyến tính phức tạp giữa đầu vào và đầu ra. Nếu không có hàm kích hoạt (tức là mạng chỉ gồm các phép biến đổi tuyến tính), dù có nhiều lớp nơ-ron đi chăng nữa, mạng cũng chỉ tương đương với một phép biến đổi tuyến tính đơn giản.

2. Giới hạn giá trị đầu ra:

Một số hàm kích hoạt như **Sigmoid** hoặc **Tanh** néo đầu ra vào một khoảng giới hạn (ví dụ: Sigmoid giới hạn từ 0 đến 1, Tanh từ -1 đến 1). Điều này giúp giữ cho các giá trị không bị tăng quá lớn hoặc nhỏ quá mức, giúp ổn định quá trình huấn luyện.

3. Giúp quá trình lan truyền ngược (Backpropagation):

Hàm kích hoạt tạo ra các đạo hàm có thể tính toán được, giúp tối ưu hóa trọng số qua quá trình lan truyền ngược. Các hàm như **ReLU** (Rectified Linear Unit) giúp giải quyết vấn đề **gradient biến mất** (vanishing gradient) mà các hàm như Sigmoid hay Tanh thường gặp phải khi mạng sâu.

4. Tạo sự phân biệt giữa các lớp (Classification):

Trong các bài toán phân loại, hàm kích hoạt ở lớp đầu ra như **Softmax** giúp chuyển đổi đầu ra thành xác suất cho từng lớp, giúp xác định lớp nào có khả năng cao nhất.

5. Kiểm soát khả năng kích hoạt của nơ-ron:

Một số hàm như **ReLU** sẽ đưa về 0 nếu đầu vào âm, giúp mạng nơ-ron trở nên thưa thớt hơn (sparse), nghĩa là chỉ một số lượng nhỏ nơ-ron được kích hoạt tại một thời điểm, giúp giảm tính phức tạp và tăng hiệu quả tính toán.

Ví dụ về các hàm kích hoạt phổ biến:

- **Sigmoid:** $\sigma(x) = \frac{1}{1 + e^{-x}}$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Tanh:** $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **ReLU:** $f(x) = \max(0, x)$

$$f(x) = \max(0, x)$$

- **Leaky ReLU:** $f(x) = x$ nếu $x > 0$, và αx nếu $x \leq 0$ với α nhỏ.

$$f(x) = x$$
 nếu $x > 0$, và αx nếu $x \leq 0$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

$$\alpha$$

- **Softmax:** Biến đổi đầu ra thành xác suất, thường dùng trong bài toán phân loại đa lớp.

Bạn đang làm việc với loại mạng nào vậy? CNN, RNN hay MLP? Tùy từng loại mạng mà việc chọn hàm kích hoạt cũng sẽ khác đấy!

4o

You said:

vẫn chưa hiểu lắm chỗ Tính phi tuyến tính , bạn có thể lấy ví dụ có và ko có hàm kích họa ko

ChatGPT said:

Để giúp bạn hiểu rõ hơn về **tính phi tuyến tính** của hàm kích hoạt, mình sẽ lấy một ví dụ đơn giản với và không có hàm kích hoạt trong mạng nơ-ron.

1. Trường hợp KHÔNG có hàm kích hoạt (Chỉ tuyến tính):

Giả sử bạn có một mạng nơ-ron đơn giản với **2 lớp** (Layer 1 và Layer 2). Nếu **không** dùng hàm kích hoạt, mỗi lớp chỉ thực hiện phép biến đổi tuyến tính (nhân ma trận và cộng).

Công thức:

$$\text{Đa}^{\wedge} \text{ ra} = W_2(W_1x + b_1) + b_2 \quad (\text{Đầu ra}) = W_2(W_1x + b_1) + b_2$$

$$\text{Đa}^{\wedge} \text{ ra} = W_2(W_1x + b_1) + b_2$$

Trong đó:

- xxx là đầu vào.
- W1,W2W_1, W_2W1,W2 là trọng số (weights) của các lớp.
- b1,b2b_1, b_2b1,b2 là các hệ số điều chỉnh (bias).

Nhận xét:

Phép nhân và cộng chỉ là **biến đổi tuyến tính**. Khi bạn kết hợp nhiều lớp tuyến tính lại với nhau, kết quả cuối cùng vẫn chỉ là một **biến đổi tuyến tính đơn giản**. Điều này có nghĩa là **mạng không thể học được các mối quan hệ phức tạp**, chẳng hạn như phân biệt các dữ liệu không tuyến tính.

2. Trường hợp CÓ hàm kích hoạt (Phi tuyến tính):

Bây giờ, nếu bạn thêm một hàm kích hoạt fff (ví dụ: ReLU hoặc Sigmoid) sau mỗi lớp, công thức sẽ như sau:

Công thức:

$$\text{Đa}^{\wedge}\text{ ra} = W_2 \cdot f(W_1x + b_1) + b_2 \quad \text{Đầu ra} = W_2 \cdot f(W_1x + b_1) + b_2$$

$$\text{Đa}^{\wedge}\text{ ra} = W_2 \cdot f(W_1x + b_1) + b_2$$

Ở đây, hàm fff **phi tuyến tính** (ReLU, Sigmoid, Tanh, v.v.) sẽ làm "bè cong" dữ liệu sau mỗi lớp.

Nhận xét:

- Nhờ có hàm kích hoạt, mạng nơ-ron có thể học các mối quan hệ **phi tuyến tính** giữa đầu vào và đầu ra.
 - Điều này giúp mạng giải quyết được các bài toán phức tạp hơn, ví dụ như phân loại các điểm dữ liệu không thể phân tách bằng một đường thẳng.
-

Ví dụ minh họa cụ thể:

Giả sử bạn có một bài toán phân loại dữ liệu sau:

- **Dữ liệu tuyến tính:** Các điểm dữ liệu có thể phân biệt bằng một đường thẳng. Không cần hàm kích hoạt cũng có thể giải quyết được.
- **Dữ liệu phi tuyến tính:** Các điểm dữ liệu sắp xếp theo hình vòng tròn hoặc xoắn ốc. Nếu **không có hàm kích hoạt**, mạng sẽ không thể học cách phân

bietet các điểm này.

Hình ảnh minh họa (giả sử):

1. **Không có hàm kích hoạt (Tuyến tính):** Mạng chỉ có thể vẽ một đường thẳng để phân loại dữ liệu.
2. **Có hàm kích hoạt (Phi tuyến tính):** Mạng có thể tạo ra các đường cong hoặc vùng phức tạp để phân loại dữ liệu.

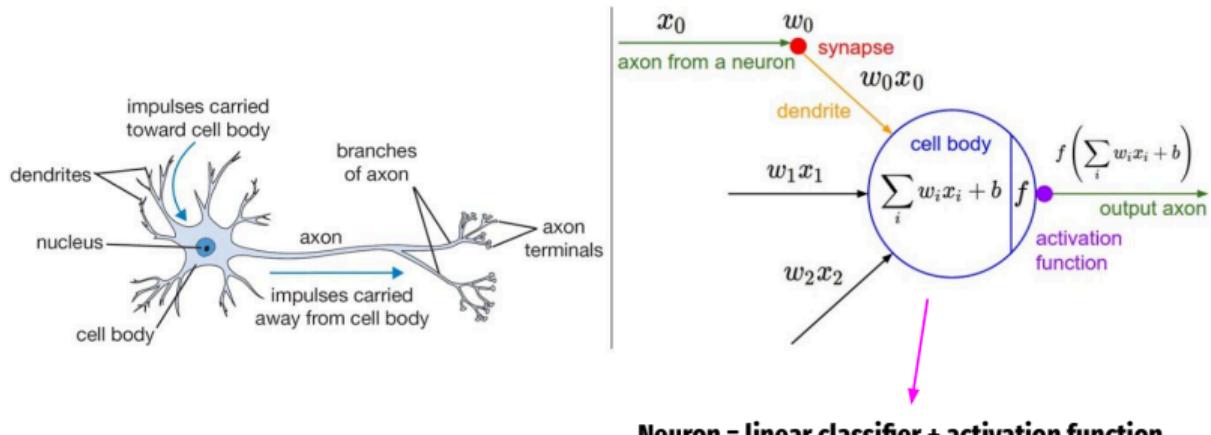
Kết luận:

- **Không có hàm kích hoạt:** Mạng chỉ giải được các bài toán đơn giản, tuyến tính.
- **Có hàm kích hoạt:** Mạng có thể giải được các bài toán phức tạp hơn, như nhận dạng hình ảnh, giọng nói, dịch ngôn ngữ, v.v.

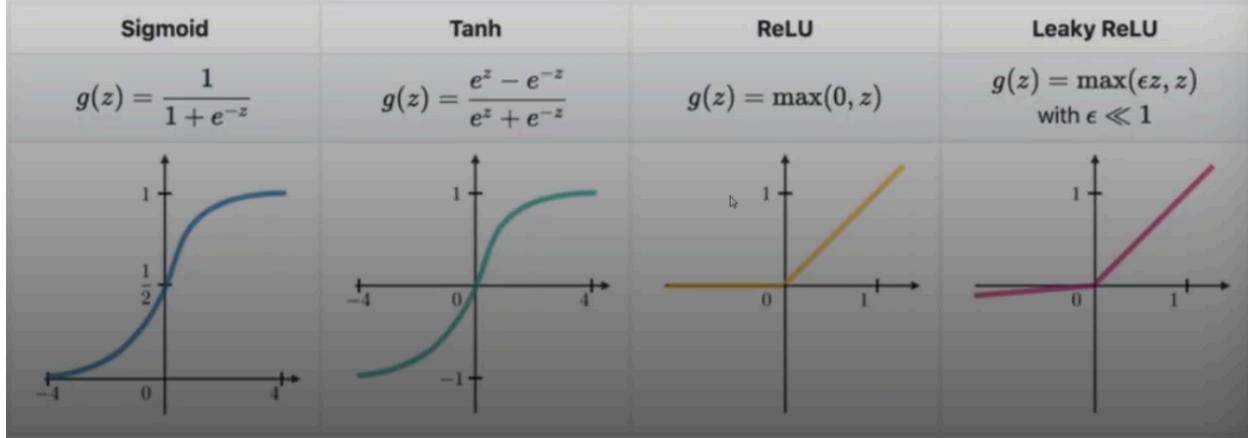
Bạn thử hình dung mạng nơ-ron như một cỗ máy. Nếu không có hàm kích hoạt, cỗ máy chỉ làm được các phép toán cơ bản, còn khi thêm hàm kích hoạt, nó có thể "sáng tạo" ra các phép biến đổi phức tạp hơn. 😊

Bạn thấy ví dụ này rõ hơn chưa? Nếu muốn mình có thể làm thêm một ví dụ bằng code nhỏ để minh họa!

Biological Neuron vs Artificial Neuron



Activation function

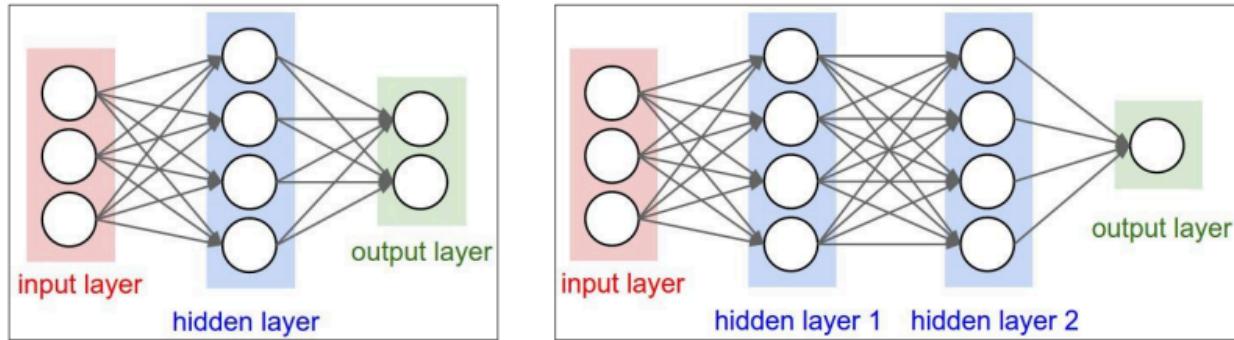


các hàm kích hoạt đều là phi tuyến tính

Sigmoid: Convert output trong range bất kỳ sang range từ 0 đến 1

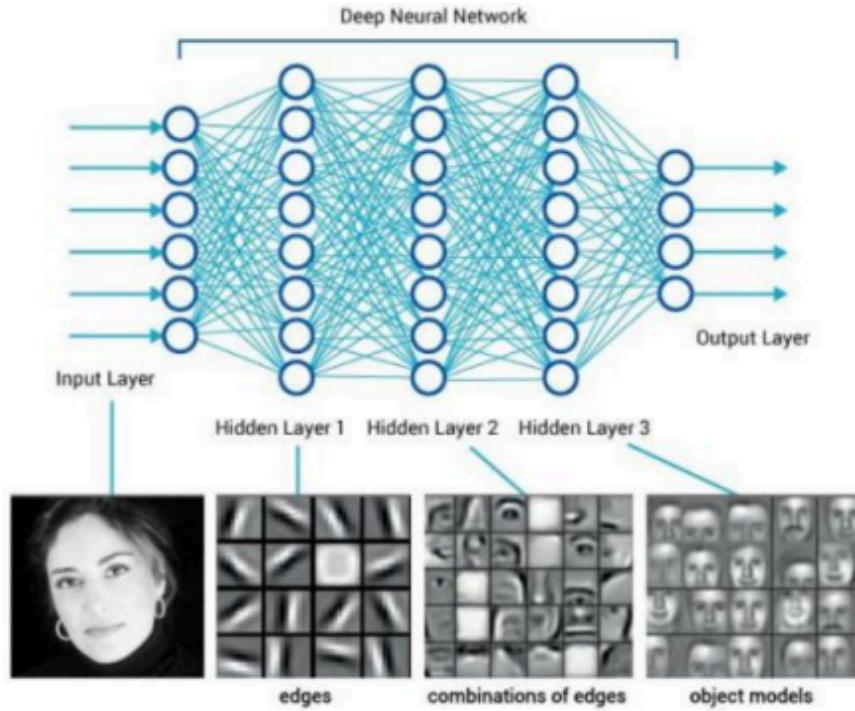
gây ra 1 hiện tượng là **gradient biến mất** (vanishing gradient)

Neural Network Architecture



Fully connectect layer

Deep Neural Network



=====

; Ượng giá model

imentionaty reduce và clustering trong học ko giám sát là gì

ChatGPT said:

Dimension Reduction (Giảm chiều dữ liệu) và Clustering (Phân cụm) trong Học không giám sát

Trong học không giám sát (*unsupervised learning*), hai kỹ thuật phổ biến là **giảm chiều dữ liệu (dimension reduction)** và **phân cụm (clustering)**. Cả hai đều được sử dụng để tìm ra cấu trúc trong dữ liệu mà không cần nhãn (label).

1. Dimension Reduction (Giảm chiều dữ liệu)

Giảm chiều dữ liệu là quá trình giảm số lượng đặc trưng (features) của dữ liệu trong khi vẫn giữ được thông tin quan trọng nhất. Mục tiêu chính là:

- **Loại bỏ nhiễu và giảm độ phức tạp** của dữ liệu.
- **Tăng tốc độ tính toán** trong các thuật toán học máy.
- **Dễ dàng trực quan hóa** dữ liệu (ví dụ: từ dữ liệu 100 chiều xuống 2D hoặc 3D để vẽ biểu đồ).

Các phương pháp phổ biến:

1. PCA (Principal Component Analysis - Phân tích thành phần chính)

- Tìm các hướng biến thiên lớn nhất trong dữ liệu và tạo ra các trục tọa độ mới (gọi là các thành phần chính - principal components).
- Dữ liệu được chiếu lên các trục này để giảm số chiều mà vẫn giữ lại nhiều thông tin nhất.

2. t-SNE (t-Distributed Stochastic Neighbor Embedding)

- Thích hợp cho **trực quan hóa dữ liệu** nhiều chiều thành không gian 2D hoặc 3D.
- Giữ lại quan hệ gần nhau giữa các điểm dữ liệu, phù hợp với dữ liệu có cấu trúc phức tạp.

3. LDA (Linear Discriminant Analysis - Phân tích biệt thức tuyến tính)

- Dùng khi có nhãn, thường áp dụng trong học có giám sát để tối ưu hóa việc phân loại.

2. Clustering (Phân cụm dữ liệu)

Phân cụm là kỹ thuật nhóm dữ liệu thành các cụm sao cho các điểm trong cùng một cụm có đặc điểm giống nhau hơn so với các điểm ở cụm khác.

Các thuật toán phân cụm phổ biến:

1. K-Means

- Chia dữ liệu thành K nhóm dựa trên khoảng cách giữa các điểm dữ liệu và các trung tâm cụm.

KK

- Lặp lại quá trình điều chỉnh trung tâm cụm cho đến khi hội tụ.
- Nhược điểm: Cần chọn trước số cụm K.

KK

2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Phân cụm dựa trên mật độ điểm dữ liệu, không cần xác định trước số cụm.
- Tốt trong việc xử lý dữ liệu có hình dạng phức tạp và phát hiện **outliers** (điểm nhiễu).

3. Hierarchical Clustering (Phân cụm phân cấp)

- Xây dựng cây phân cấp của các cụm mà không cần xác định trước số cụm.
- Có thể chia nhỏ thành hai loại chính:
 - **Agglomerative (từ dưới lên)**: Bắt đầu với mỗi điểm dữ liệu là một cụm riêng lẻ, sau đó gộp dần.
 - **Divisive (từ trên xuống)**: Bắt đầu với một cụm lớn, sau đó chia nhỏ dần.

Mối quan hệ giữa Dimension Reduction và Clustering

- Giảm chiều dữ liệu giúp phân cụm hiệu quả hơn, vì dữ liệu có ít chiều hơn sẽ giảm nhiễu và tính toán nhanh hơn.
- Sau khi giảm chiều dữ liệu, có thể áp dụng thuật toán phân cụm như K-Means hoặc DBSCAN để nhóm dữ liệu.
- Ví dụ:
 - Dùng **PCA** để giảm dữ liệu từ 100 chiều xuống 2D, rồi dùng **K-Means** để phân cụm.
 - Dùng **t-SNE** để trực quan hóa dữ liệu trước khi phân cụm bằng **DBSCAN**.

Kết luận

- **Dimension Reduction** giúp giảm kích thước dữ liệu, loại bỏ nhiễu và tăng tốc thuật toán.

- **Clustering** giúp tìm ra nhóm dữ liệu có đặc điểm giống nhau mà không cần nhãn.
- Kết hợp cả hai sẽ giúp phân tích dữ liệu hiệu quả hơn trong học không giám sát.

40

You said:

giữa cluster và classifi khác nhau chồ nà

ChatGPT said:

Sự khác nhau giữa Clustering và Classification

1. Clustering (Phân cụm - Học không giám sát)

- **Là gì?**
 - Là kỹ thuật trong học không giám sát (*unsupervised learning*), dùng để nhóm dữ liệu vào các cụm (*clusters*) dựa trên sự tương đồng giữa chúng.
 - Không có nhãn (*labels*) sẵn, thuật toán phải tự tìm ra các nhóm trong dữ liệu.
- **Mục tiêu:**
 - Tìm ra cấu trúc ẩn trong dữ liệu.
 - Nhóm các đối tượng có tính chất giống nhau vào cùng một cụm.
- **Ví dụ thực tế:**
 - Nhóm khách hàng thành các nhóm dựa trên hành vi mua sắm.
 - Phân cụm tin tức thành các chủ đề khác nhau.
 - Nhận diện các loại tế bào ung thư dựa trên đặc điểm sinh học.
- **Các thuật toán phổ biến:**
 - **K-Means**
 - **DBSCAN**
 - **Hierarchical Clustering**

2. Classification (Phân loại - Học có giám sát)

- **Là gì?**

- Là kỹ thuật trong học có giám sát (*supervised learning*), dùng để dự đoán nhãn (*label*) của một đối tượng dựa trên dữ liệu huấn luyện đã có nhãn trước đó.

- **Mục tiêu:**

- Xác định một đối tượng thuộc vào nhóm nào trong tập hợp các nhóm có sẵn.

- **Ví dụ thực tế:**

- Phân loại email là *spam* hay *không spam*.
- Nhận diện chữ số viết tay (0-9).
- Dự đoán một bệnh nhân có bị ung thư hay không dựa trên dữ liệu y tế.

- **Các thuật toán phổ biến:**

- **Decision Tree (Cây quyết định)**
- **Random Forest**
- **SVM (Support Vector Machine)**
- **Naive Bayes**
- **Neural Networks (Mạng nơ-ron nhân tạo)**

3. Điểm khác nhau giữa Clustering và Classification

Tiêu chí	Clustering (Phân cụm)	Classification (Phân loại)
Loại học máy	Học không giám sát (<i>Unsupervised Learning</i>)	Học có giám sát (<i>Supervised Learning</i>)
Dữ liệu huấn luyện	Không có nhãn (chỉ có dữ liệu đầu vào)	Có nhãn (có cả dữ liệu đầu vào và nhãn đầu ra)
Mục tiêu	Nhóm dữ liệu thành các cụm dựa trên sự tương đồng	Xác định lớp (<i>class</i>) của một dữ liệu mới dựa trên mẫu đã học
Ví dụ	Phân nhóm khách hàng thành các cụm	Dự đoán khách hàng có mua sản phẩm hay không

Đầu ra	Các cụm không có nhãn rõ ràng	Các lớp có nhãn xác định (ví dụ: "Spam" hoặc "Not Spam")
Thuật toán phổ biến	K-Means, DBSCAN, Hierarchical Clustering	SVM, Random Forest, Decision Tree, Neural Networks

4. Kết luận

- **Clustering:** Dùng để tìm ra các nhóm tự nhiên trong dữ liệu mà không có nhãn sẵn.
- **Classification:** Dùng để gán nhãn cho dữ liệu mới dựa trên dữ liệu huấn luyện có sẵn.
- **Khi nào dùng cái nào?**
 - Nếu dữ liệu có nhãn → **Classification**.
 - Nếu dữ liệu chưa có nhãn và cần tìm cấu trúc → **Clustering**.
- **Ví dụ kết hợp cả hai:**
 - Dùng **Clustering** để nhóm khách hàng thành các cụm, sau đó gán nhãn cho các cụm này và dùng **Classification** để phân loại khách hàng mới vào cụm thích hợp.

4o

You said:

việc đánh giá 1 model rồi thì loss function, bias... có liên quan với nhau ko

ChatGPT said:

Mối liên hệ giữa Loss Function, Bias, và các yếu tố đánh giá Model

Khi đánh giá một mô hình học máy, các khái niệm như **Loss Function (hàm mất mát)**, **Bias (độ chệch)**, **Variance (phương sai)**, **Overfitting (quá khớp)**, **Underfitting (chưa khớp)** có mối quan hệ chặt chẽ với nhau. Dưới đây là cách chúng liên quan đến nhau:

1. Loss Function (Hàm mất mát)

- **Là gì?**
 - Hàm đo lường mức độ sai lệch giữa giá trị dự đoán của mô hình và giá trị thực tế.
 - Loss Function càng nhỏ thì mô hình càng chính xác.
 - **Các loại Loss Function phổ biến:**
 - **Regression (Hồi quy):**
 - Mean Squared Error (MSE)
 - Mean Absolute Error (MAE)
 - **Classification (Phân loại):**
 - Cross-Entropy Loss
 - Hinge Loss (cho SVM)
 - **Liên quan đến Bias và Variance:**
 - Nếu Loss Function quá cao trên tập huấn luyện → **Underfitting (Bias cao).**
 - Nếu Loss Function rất thấp trên tập huấn luyện nhưng cao trên tập kiểm tra → **Overfitting (Variance cao).**
-

2. Bias (Độ chêch)

- **Là gì?**
 - Bias thể hiện mức độ mà mô hình đơn giản hóa quá mức dữ liệu, dẫn đến dự đoán kém chính xác.
 - **Bias cao** → Mô hình không học được tốt từ dữ liệu huấn luyện (**underfitting**).
 - **Bias thấp** → Mô hình có thể học tốt từ dữ liệu huấn luyện.
- **Ví dụ:**
 - Dùng mô hình đường thẳng (hồi quy tuyến tính) để dự đoán dữ liệu có dạng parabol sẽ dẫn đến bias cao.
- **Liên quan đến Loss Function:**

- Nếu Bias cao, Loss Function trên tập huấn luyện cũng sẽ cao.
-

3. Variance (Phương sai)

- **Là gì?**
 - Đo lường mức độ mà mô hình thay đổi khi huấn luyện trên các tập dữ liệu khác nhau.
 - **Variance cao** → Mô hình quá nhạy cảm với dữ liệu huấn luyện (**overfitting**).
 - **Variance thấp** → Mô hình tổng quát hóa tốt hơn.
 - **Ví dụ:**
 - Mô hình quá phức tạp như deep neural networks có thể có variance cao, làm cho nó hoạt động rất tốt trên tập huấn luyện nhưng kém trên tập kiểm tra.
 - **Liên quan đến Loss Function:**
 - Nếu Variance cao, Loss Function trên tập huấn luyện sẽ rất thấp, nhưng trên tập kiểm tra sẽ cao.
-

4. Mối quan hệ giữa Loss Function, Bias và Variance

- Có thể tóm tắt bằng **Trade-off Bias-Variance**:

Bias	Variance	Loss Function (Train Set)	Loss Function (Test Set)	Kết quả
Cao	Thấp	Cao	Cao	Underfitting
Thấp	Cao	Thấp	Cao	Overfitting
Trung bình	Trung bình	Trung bình	Trung bình	Tốt nhất (Generalization)

- Khi huấn luyện mô hình:
 - **Bias cao & Variance thấp** → Mô hình đơn giản, ít học được từ dữ liệu (**underfitting**).
 - **Bias thấp & Variance cao** → Mô hình quá phức tạp, học cả nhiễu (**overfitting**).

- Bias vừa đủ & Variance vừa đủ → Mô hình tốt nhất, tổng quát hóa tốt.

5. Cách điều chỉnh Bias và Variance

Giải pháp	Giảm Bias	Giảm Variance
Thêm nhiều đặc trưng (features)	✓	✗
Thêm nhiều dữ liệu	✗	✓
Dùng mô hình phức tạp hơn (Deep Learning, Random Forest)	✓	✗
Dùng mô hình đơn giản hơn (Linear Regression, Decision Tree nhỏ)	✗	✓
Thêm Regularization (L1, L2)	✗	✓
Data Augmentation	✗	✓

6. Kết luận

- Loss Function đo lường lỗi của mô hình, nhưng chỉ số loss thấp không có nghĩa là mô hình tốt.
- Bias cao dẫn đến underfitting, variance cao dẫn đến overfitting.
- Mục tiêu là tìm sự cân bằng giữa bias và variance để mô hình có khả năng tổng quát hóa tốt nhất.

40

You said:

khi model ko tốt thì sẽ làm gì tiếp

ChatGPT said:

Khi mô hình không tốt, có nhiều cách để cải thiện tùy thuộc vào nguyên nhân gây ra vấn đề. Dưới đây là các bước quan trọng để khắc phục:

1. Xác định nguyên nhân chính của vấn đề

Trước khi điều chỉnh mô hình, cần hiểu tại sao nó hoạt động kém. Kiểm tra các yếu tố sau:

- **Loss Function cao trên cả tập train và test → Underfitting (Bias cao).**
 - **Loss Function thấp trên train nhưng cao trên test → Overfitting (Variance cao).**
 - **Dữ liệu có nhiễu hoặc không đủ thông tin → Dữ liệu không đủ tốt để mô hình học.**
-

2. Cách khắc phục khi mô hình bị Underfitting (Bias cao)

Dấu hiệu

- Loss Function trên tập huấn luyện cao.
- Accuracy (độ chính xác) thấp trên cả train và test.

Giải pháp

Dùng mô hình phức tạp hơn

- Chuyển từ Linear Regression → Polynomial Regression.
- Chuyển từ Logistic Regression → Neural Network.

Thêm đặc trưng (Feature Engineering)

- Thêm các feature mới giúp mô hình học tốt hơn.
- Dùng PCA hoặc Feature Selection để chọn đặc trưng quan trọng.

Tăng số lượng Epochs (cho Deep Learning)

- Train mô hình lâu hơn để học được nhiều hơn.

Dùng thuật toán tốt hơn

- Với dữ liệu dạng bảng, thử **Random Forest hoặc XGBoost** thay vì Linear Regression.
- Với dữ liệu ảnh, thử **CNN thay vì MLP**.

Thu thập thêm dữ liệu

- Nếu dữ liệu quá ít, mô hình sẽ khó học được quy luật.
-

3. Cách khắc phục khi mô hình bị Overfitting (Variance cao)

Dấu hiệu

- Loss Function trên train thấp, nhưng trên test rất cao.
- Accuracy trên train cao, nhưng trên test thấp.

Giải pháp

Thu thập thêm dữ liệu

- Càng nhiều dữ liệu, mô hình càng tổng quát hóa tốt hơn.

Dùng Regularization (L1, L2)

- Thêm **L1 (Lasso)** hoặc **L2 (Ridge)** vào loss function để hạn chế overfitting.

Giảm số lượng đặc trưng (Feature Selection)

- Nếu có quá nhiều feature không quan trọng, có thể gây overfitting.
- Dùng PCA hoặc chọn feature quan trọng nhất.

Sử dụng Dropout (cho Deep Learning)

- Trong mạng nơ-ron, thêm Dropout layer để tránh quá khớp.

Dùng kỹ thuật Data Augmentation

- Nếu làm việc với ảnh, có thể xoay, lật, thay đổi màu sắc hình ảnh để tăng dữ liệu.

Dùng mô hình đơn giản hơn

- Nếu đang dùng Deep Neural Network quá phức tạp, thử dùng mô hình đơn giản hơn như Random Forest hoặc SVM.

Early Stopping

- Nếu loss trên train tiếp tục giảm nhưng loss trên test tăng, dừng sớm quá trình train để tránh overfitting.
-

4. Kiểm tra và tối ưu hóa dữ liệu

Dữ liệu là yếu tố quan trọng nhất quyết định chất lượng của mô hình.

Giải pháp

Làm sạch dữ liệu

- Loại bỏ dữ liệu bị thiếu, nhiễu, hoặc không hợp lệ.

Chuẩn hóa dữ liệu (Normalization/Standardization)

- Với các thuật toán như SVM, Neural Networks, cần chuẩn hóa dữ liệu về cùng một thang đo.

Xử lý dữ liệu mất cân bằng (Class Imbalance)

- Nếu dữ liệu bị mất cân bằng (vd: 90% là lớp A, 10% là lớp B), có thể dùng **SMOTE** hoặc **re-sampling** để cân bằng.

Chia tập train/test đúng cách

- Đảm bảo dữ liệu train và test được chia hợp lý, không bị trùng lặp.

5. Cải thiện quá trình huấn luyện mô hình

Dùng Hyperparameter Tuning

- Thử nghiệm các giá trị khác nhau cho tham số như:
 - Learning rate
 - Batch size
 - Number of hidden layers (cho Neural Networks)
 - Number of trees (cho Random Forest, XGBoost)
- Dùng **Grid Search** hoặc **Bayesian Optimization** để tìm giá trị tốt nhất.

Sử dụng Ensemble Learning

- Kết hợp nhiều mô hình khác nhau để có kết quả tốt hơn (Bagging, Boosting).
- Ví dụ: **Random Forest**, **XGBoost**, **LightGBM**.

Chia nhỏ tập train (Cross-validation)

- Dùng **K-Fold Cross Validation** để đảm bảo mô hình hoạt động tốt trên nhiều tập dữ liệu khác nhau.
-

6. Thử nghiệm lại mô hình và đánh giá

- Sau khi áp dụng các kỹ thuật trên, cần kiểm tra lại:
 - Loss Function có giảm không?**
 - Độ chính xác trên tập test có cải thiện không?**
 - Bias và Variance có cân bằng chưa?**
 - Nếu vẫn chưa tốt, có thể cần **quay lại và thử nghiệm cách tiếp cận khác**.
-

Tóm tắt cách cải thiện mô hình kém

Vấn đề	Nguyên nhân	Cách khắc phục
Underfitting (Bias cao)	Mô hình quá đơn giản, chưa học đủ	Thêm feature, dùng mô hình phức tạp hơn, train lâu hơn
Overfitting (Variance cao)	Mô hình quá phức tạp, học cả nhiễu	Thêm dữ liệu, Regularization, Dropout, giảm số feature
Dữ liệu kém	Nhiều, mất cân bằng, chưa chuẩn hóa	Làm sạch dữ liệu, cân bằng dữ liệu, chuẩn hóa dữ liệu
Hyperparameters chưa tối ưu	Học quá nhanh/chậm, batch size sai	Dùng Grid Search, thử các giá trị khác nhau
Không kiểm tra tốt	Chia tập train/test sai, đánh giá chưa kỹ	K-Fold Cross Validation, kiểm tra Loss trên test

Kết luận

Nếu mô hình hoạt động kém, hãy làm theo các bước sau:

- 1 **Kiểm tra nguyên nhân (Bias, Variance, Dữ liệu, Hyperparameters, v.v.).**
- 2 **Chọn giải pháp phù hợp** để cải thiện.
- 3 **Tối ưu hóa mô hình** bằng Regularization, Feature Engineering, Data Augmentation.

4 Thủ nghiệm lại và đánh giá để đảm bảo mô hình tốt hơn.

Nếu thử nhiều cách nhưng vẫn không đạt kết quả mong muốn, có thể xem xét lại **bài toán gốc** hoặc **chọn mô hình khác**. 🚀

4o

You said:

vậy việc cập nhật lại trọng số của mô hình ở bước nào

ChatGPT said:

Việc **cập nhật lại trọng số của mô hình** xảy ra trong **quá trình huấn luyện (training phase)**, cụ thể là trong bước **Lan truyền ngược (Backpropagation) và Tối ưu hóa (Optimization)**.

1. Quá trình cập nhật trọng số của mô hình diễn ra ở đâu?

Cập nhật trọng số diễn ra **sau mỗi vòng lặp huấn luyện (epoch hoặc batch)** bằng cách sử dụng thuật toán tối ưu hóa như **Gradient Descent**.

2. Các bước chi tiết của quá trình cập nhật trọng số

Mô hình học máy (ML) hoặc mạng nơ-ron (Neural Network) thường cập nhật trọng số theo quy trình sau:

Bước 1: Truyền xuôi (Forward Propagation)

- Dữ liệu đầu vào X đi qua các lớp của mô hình.
 XX
- Mỗi lớp có **trọng số WWW** và **hệ số dịch bbb** (bias).
- Dự đoán \hat{Y} được tính toán.
 $\hat{Y} \mid \hat{Y}$

Bước 2: Tính hàm mất mát (Loss Function)

- So sánh dự đoán \hat{Y} với giá trị thực Y .

$\hat{Y} \mid \hat{Y}$

YY

- Dùng hàm mất mát (Loss Function) như MSE (Mean Squared Error) hoặc Cross-Entropy để tính **mức độ sai số**.

Bước 3: Lan truyền ngược (Backpropagation)

- Tính **đạo hàm của Loss Function theo từng trọng số** bằng cách áp dụng **đạo hàm chuỗi (Chain Rule)**.
- Công thức:

$$\partial W \partial L = \partial Y^T \partial L \cdot \partial W \partial Y^T$$

$$\partial L \partial W = \partial L \partial Y^T \cdot \partial Y^T \partial W \frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial W}$$
- Đạo hàm này cho biết **trọng số nào cần giảm/tăng** để giảm loss.

Bước 4: Cập nhật trọng số bằng thuật toán tối ưu hóa

- Dùng Gradient Descent hoặc các biến thể như **Adam, RMSprop, SGD** để cập nhật trọng số:

$$W_{\text{new}} = W_{\text{old}} - \alpha \cdot \partial W \partial L$$

$$W_{\text{new}} = W_{\text{old}} - \alpha \cdot \partial L \partial W = W_{\text{old}} - \alpha \cdot \frac{\partial L}{\partial W}$$
 - α là **learning rate** (tốc độ học).
 - Nếu đạo hàm lớn → giảm nhiều.
 - Nếu đạo hàm nhỏ → giảm ít.

Bước 5: Lặp lại quá trình cho đến khi loss nhỏ nhất

- Cứ mỗi epoch, trọng số tiếp tục được cập nhật để mô hình dần dần tối ưu.

3. Khi nào cần cập nhật lại trọng số?

Việc cập nhật trọng số diễn ra **mỗi lần lan truyền ngược**, cụ thể:

- Mỗi batch (Mini-Batch Gradient Descent)**
- Mỗi epoch (Full-Batch Gradient Descent)**
- Sau khi fine-tuning mô hình (Transfer Learning, Pretrained Model)**

4. Khi mô hình không tốt, trọng số có cần thay đổi không?

- ◆ Nếu mô hình đang underfitting → Cần giảm regularization, thêm layer, hoặc tăng số epoch để cập nhật trọng số nhiều hơn.
- ◆ Nếu mô hình overfitting → Có thể thêm dropout, giảm số layer, hoặc điều chỉnh learning rate để tránh cập nhật quá mạnh.
- ◆ Nếu loss không giảm → Điều chỉnh learning rate hoặc chọn thuật toán tối ưu hóa khác.

5. Tổng kết

- Trọng số được cập nhật liên tục trong quá trình huấn luyện thông qua Backpropagation + Gradient Descent.
- Cập nhật diễn ra sau mỗi batch hoặc epoch để giúp mô hình giảm lỗi.
- Nếu mô hình không hoạt động tốt, cần điều chỉnh hàm mất mát, learning rate, số epoch, regularization, v.v. để cập nhật trọng số hiệu quả hơn.

👉 Nếu bạn muốn tối ưu hóa việc cập nhật trọng số, hãy thử **Adam Optimizer** vì nó giúp điều chỉnh learning rate tự động! 🚀

40

You said:

Loss Function dùng đánh giá mà

ChatGPT said:

Đúng rồi! **Loss Function (hàm mất mát)** được dùng để **đánh giá độ chính xác của mô hình** trong quá trình huấn luyện. Tuy nhiên, nó cũng liên quan trực tiếp đến việc **cập nhật trọng số** thông qua thuật toán tối ưu hóa.

1. Loss Function có nhiệm vụ gì?

Loss Function đo lường **mức độ sai số giữa dự đoán \hat{Y} và giá trị thực Y** . Mục tiêu là làm cho loss càng nhỏ càng tốt.

Ví dụ:

- Nếu mô hình dự đoán giá nhà là **\$150,000**, nhưng thực tế là **\$200,000**, thì loss sẽ cao.
 - Nếu mô hình dự đoán gần đúng, ví dụ **\$198,000**, thì loss sẽ thấp.
-

2. Loss Function có ảnh hưởng đến việc cập nhật trọng số không?

 **Có!** Trọng số WWW được cập nhật dựa trên **đạo hàm của Loss Function**.

Công thức:

$$W_{\text{new}} = W_{\text{old}} - \alpha \cdot \partial L / \partial W$$

$$W_{\text{new}} = W_{\text{old}} - \alpha \cdot \nabla L$$

- $\partial L / \partial W$ là **gradient của Loss Function theo trọng số**.
- α là **learning rate**, quyết định tốc độ cập nhật trọng số.
- Nếu loss lớn \rightarrow trọng số thay đổi mạnh để giảm loss.
- Nếu loss nhỏ \rightarrow trọng số thay đổi ít.

◆ **Tóm lại:** Loss Function giúp đánh giá mô hình, nhưng nó cũng ảnh hưởng trực tiếp đến quá trình tối ưu hóa, vì trọng số thay đổi dựa trên giá trị loss.

3. Loss Function khác gì so với Evaluation Metrics?

- **Loss Function:** Dùng để **tối ưu mô hình** trong training (có gradient, giúp cập nhật trọng số).
- **Evaluation Metrics:** Dùng để **đánh giá mô hình** sau khi train xong (không ảnh hưởng đến trọng số).

◆ **Ví dụ:**

- **Loss Function** (có gradient, dùng để train):
 - MSE (Mean Squared Error) – cho bài toán hồi quy.
 - Cross-Entropy – cho bài toán phân loại.
- **Evaluation Metrics** (chỉ đánh giá):

- Accuracy (độ chính xác).
- Precision, Recall, F1-score.

4. Khi Loss Function thấp, mô hình có tốt không?

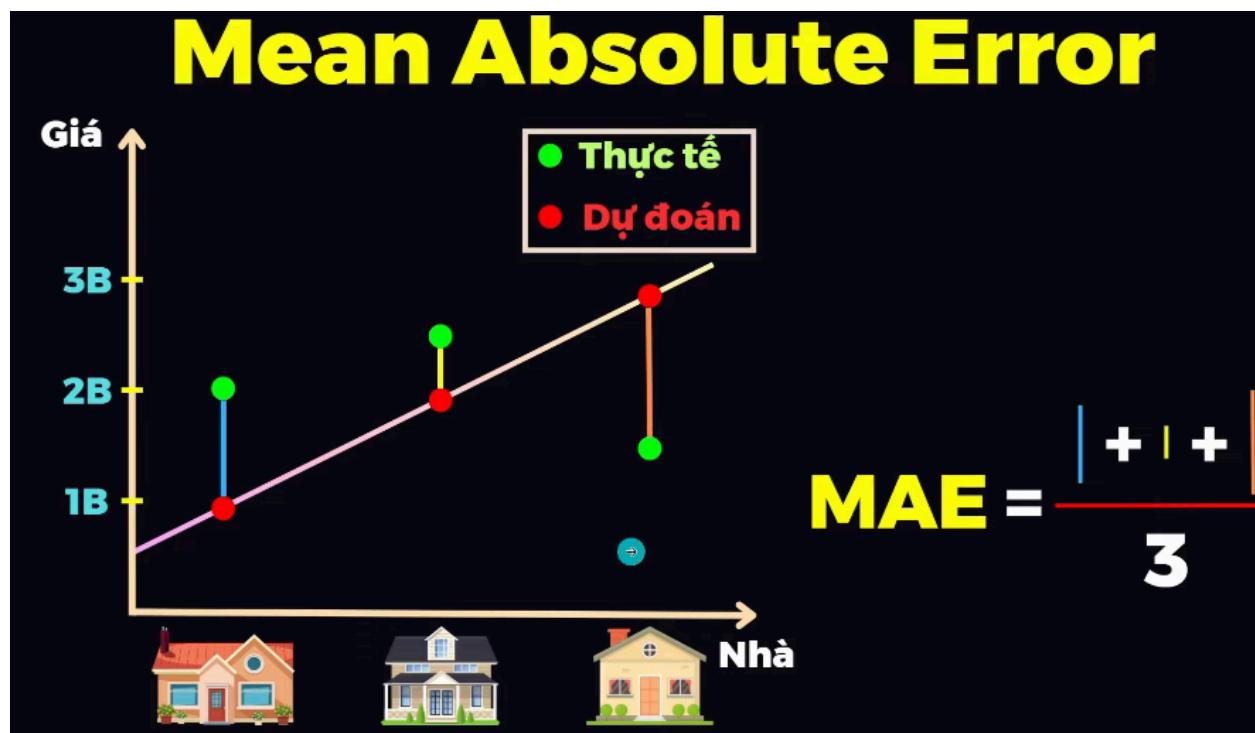
◆ **Chưa chắc!** Vì loss chỉ đo lường **mức độ sai số trung bình**, nhưng không phản ánh toàn bộ hiệu suất mô hình.

- Nếu loss thấp nhưng accuracy cũng thấp → Mô hình có thể **underfitting**.
- Nếu loss giảm nhưng validation loss tăng → **Overfitting**.

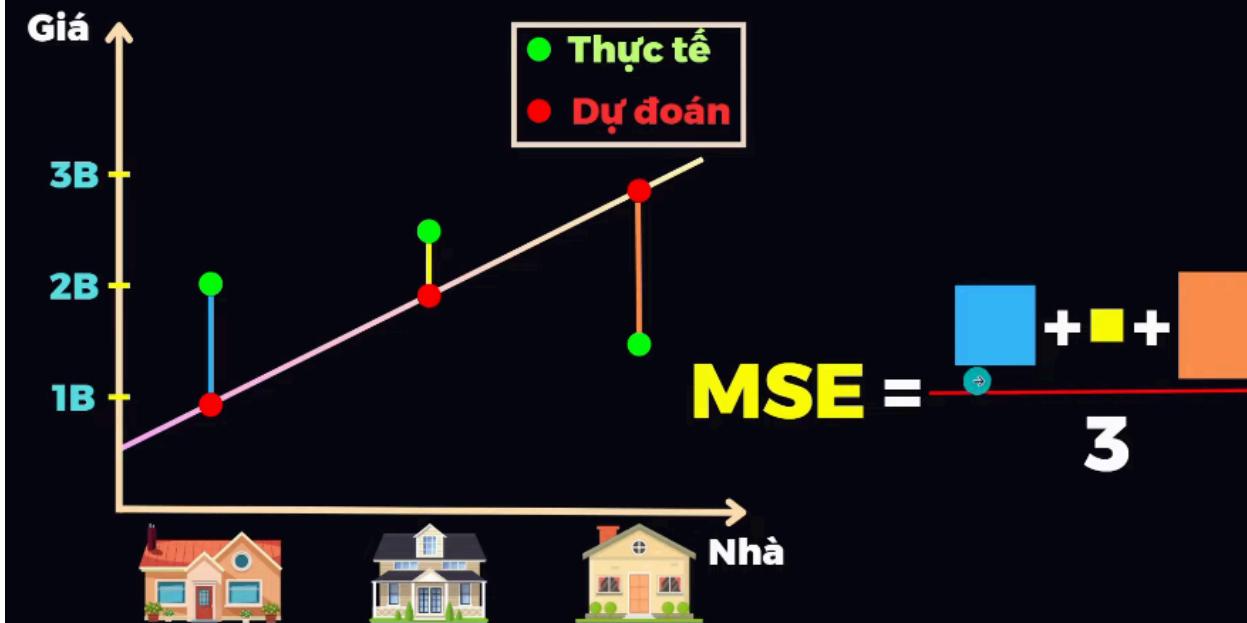
✓ Cách tốt nhất: **Kết hợp Loss Function + Evaluation Metrics để đánh giá toàn diện!** 🚀

Hồi quy

3 matrix: MSE (Mean Squared Error)



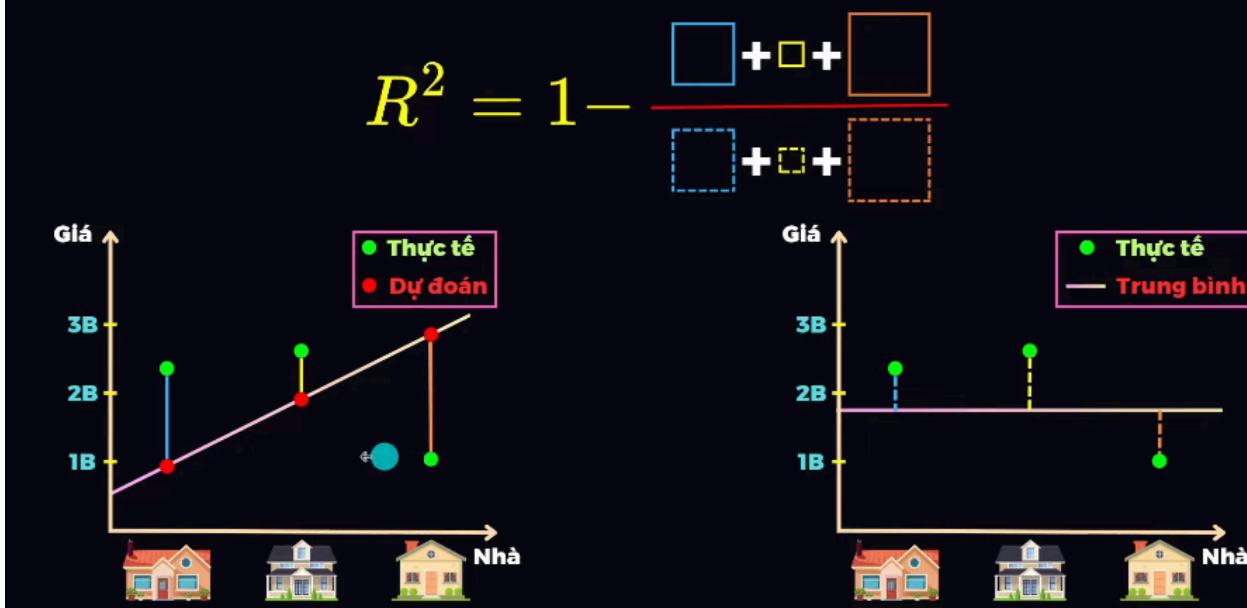
Mean Squared Error



⇒ taapj trung vào nhiều sai số lớn

thực sự khó khi nhìn vòe MSE và MAE để biết mô hình xấu hay tốt ⇒ cần dãi giá trị thực tế của đôi tượng cần dự đoán

Coefficient of Determination



Đánh giá mô hình học không giám sát (unsupervised learning) thường khó hơn so với học có giám sát vì không có nhãn để so sánh trực tiếp. Dưới đây là một số cách phổ biến để đánh giá mô hình học không giám sát tùy theo loại bài toán:

1. Đánh giá mô hình phân cụm (Clustering)

- **Silhouette Score**: Đo độ chặt chẽ của cụm và mức độ khác biệt giữa các cụm. Giá trị nằm trong khoảng [-1, 1], càng cao càng tốt.
- **Dunn Index**: Tính toán tỷ lệ giữa khoảng cách nhỏ nhất giữa các cụm và khoảng cách lớn nhất trong một cụm.
- **Davies-Bouldin Index (DBI)**: Đo mức độ tương đồng giữa các cụm, càng thấp càng tốt.
- **Elbow Method**: Xác định số cụm tối ưu bằng cách vẽ biểu đồ SSE (Sum of Squared Errors) theo số cụm.
- **Rand Index / Adjusted Rand Index (ARI)**: Nếu có nhãn thực tế, có thể dùng ARI để đo độ tương đồng giữa phân cụm và nhãn thực tế.

2. Đánh giá mô hình giảm chiều (Dimensionality Reduction)

- **Explained Variance Ratio (EVR)**: Đối với PCA, đánh giá phần trăm phương sai được giữ lại sau khi giảm chiều dữ liệu.
- **Reconstruction Error**: Đo lỗi tái tạo khi sử dụng mô hình như Autoencoder để giảm chiều dữ liệu.
- **Visualization**: Vẽ đồ thị 2D/3D bằng t-SNE hoặc UMAP để xem dữ liệu có được phân tách tốt không.

3. Đánh giá mô hình học luật liên kết (Association Rule Mining)

- **Support**: Độ phổ biến của một luật trong tập dữ liệu.
- **Confidence**: Xác suất xảy ra của phần hậu quả khi phần tiền đề xảy ra.
- **Lift**: Đánh giá mức độ hữu ích của luật, nếu >1 nghĩa là luật có ý nghĩa.

4. Đánh giá mô hình phát hiện bất thường (Anomaly Detection)

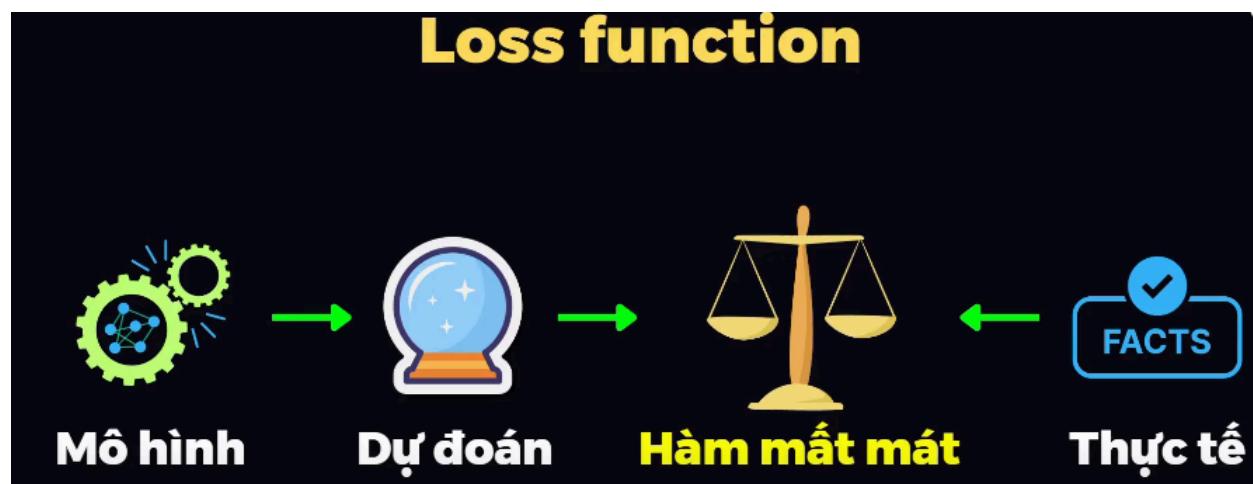
- **Reconstruction Error**: Nếu dùng Autoencoder, có thể đo lỗi tái tạo giữa dữ liệu gốc và dữ liệu tái tạo.

- **AUC-ROC / AUC-PR:** Nếu có một tập nhỏ dữ liệu được gán nhãn bất thường, có thể sử dụng đường cong ROC hoặc PR để đánh giá.
- **Z-Score / MAD (Median Absolute Deviation):** Kiểm tra xem điểm bất thường có nằm ngoài một số độ lệch chuẩn hoặc khoảng cách trung vị không.

Nếu bạn đang làm một bài toán cụ thể, mình có thể tư vấn phương pháp đánh giá phù hợp hơn.

Lost function - hàm mất mát

là thước đo xem mô hình dự đoán tốt như thế nào so với nhãn thực tế \Rightarrow cho biết mức độ sai lệch giữa giá trị thực tế và dự đoán để mô hình có thể cập nhật các tham số



các hàm loss phổ biến

Loss function

Mean absolute error

Mean squared error

$$MAE = \frac{1}{n} \sum \left| \underbrace{y - \hat{y}}_{\text{The absolute value of the residual}} \right|$$

Divide by the total number of data points
Actual output value
Predicted output value
Sum of
The absolute value of the residual

$$\text{Mean Error Squared} \\ MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Cross Entropy Loss

$$-\sum_{j=1}^M y_j \log(p(y_j))$$

Indicator variable
Prob of class j

Sum over trials

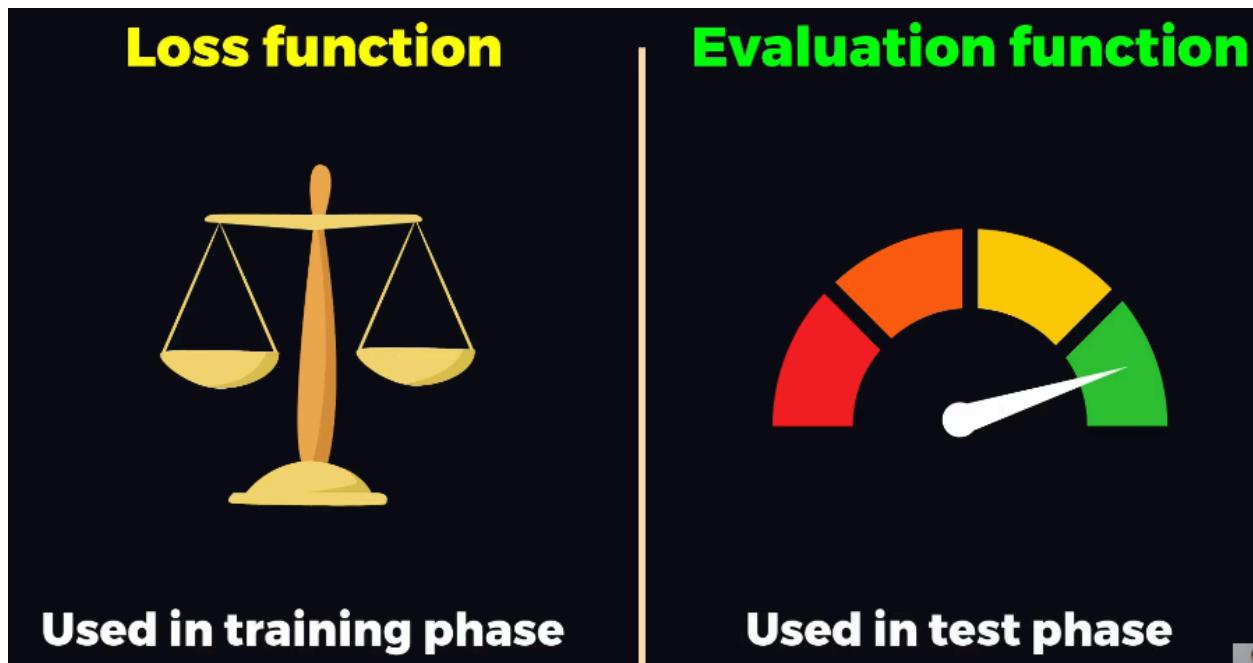
$$-\sum_{i=1}^N y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i))$$

Sum over classes

Label
Prob of positive class
Label
Prob of positive class

Cho các bài toán phân loại, cơ bản là phạt mạnh dự đoán sai

Evaluation fuinction - hàm đánh giá sẽ tạo ra các evaluation metrics lag giá trị để sử dụng để đánh giá mô hình đó tốt hay ko

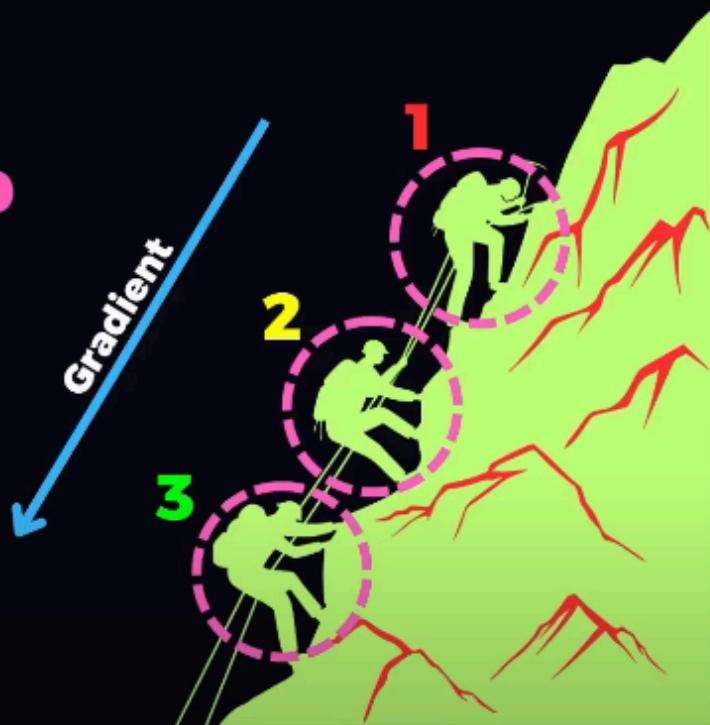


Optimazation function: hàm tối ưu hóa

nó thường kết hợp bới lost để tối ưu hóa mo hình. Nó giúp tối ưu hóa việc cập nhật tham số mô hình giúp giảm lost function

Optimization func

- SGD
- RMSprop
- AdaGrad
- Adam



Hàm mất mát

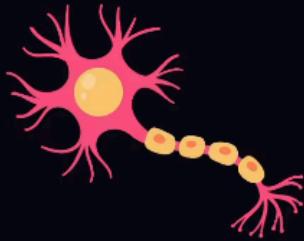
activation function - Thường dùng trong deep learning

Hàm tuyến tính giúp học và trích xuất những đặc trưng đơn giản

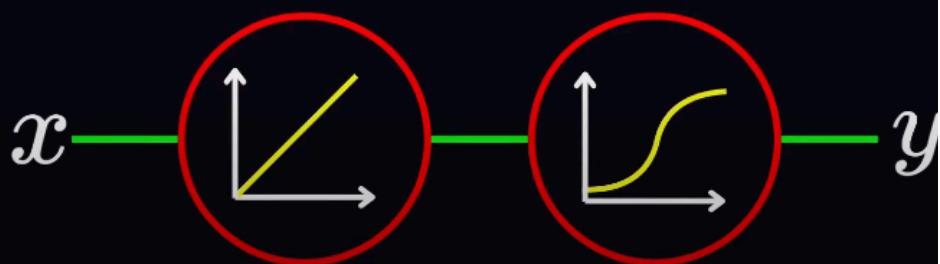
Activefunction: hàm phi tuyến \Rightarrow đặc trưng phức tạp

Activation function

Nơ-ron
sinh học

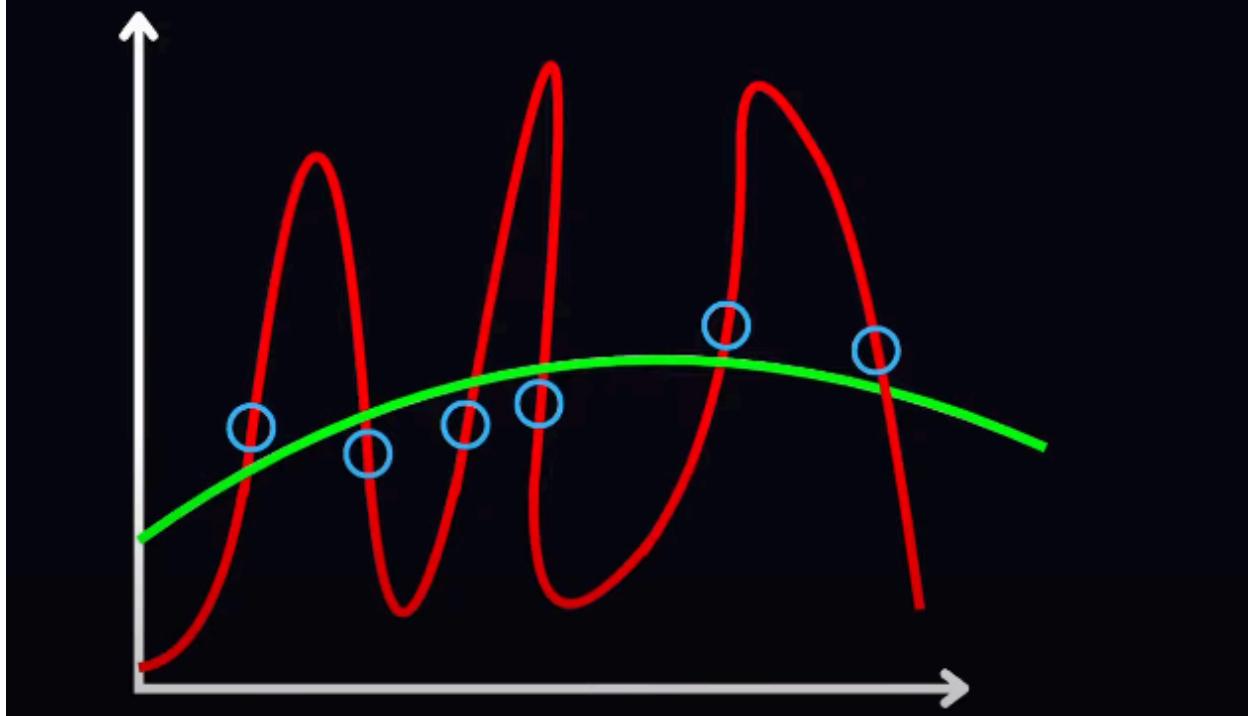


Nơ-ron
nhân tạo



Regularization function: Hmà chính quy hóa là kỹ thuật ngăn chặn overfitting bằng cách thêm các ràng buộc vào lost function để làm giảm độ phức tạp của mô hình

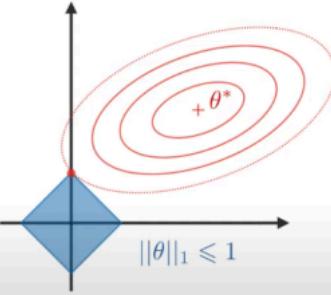
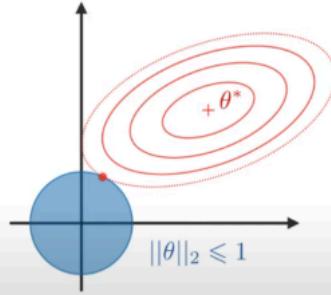
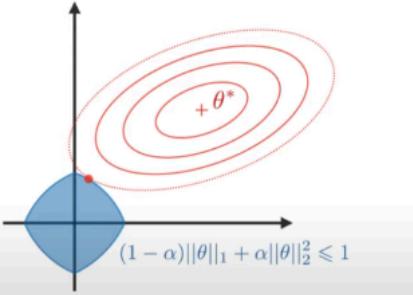
Regularization function



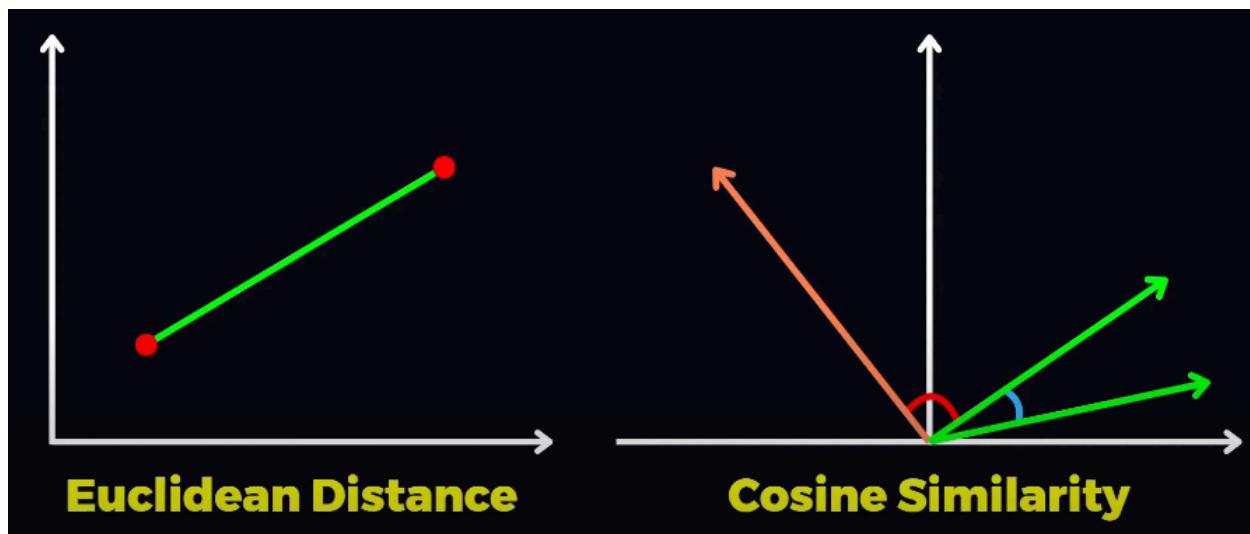
Cost Function = Loss + L2 Weight Penalty

$$= \underbrace{\sum_{i=1}^M (y_i - \sum_{j=1}^N x_{ij}w_j)^2}_{\text{Squared Error}} + \lambda \underbrace{\sum_{j=1}^N w_j^2}_{\text{L2 Regularization Term}}$$

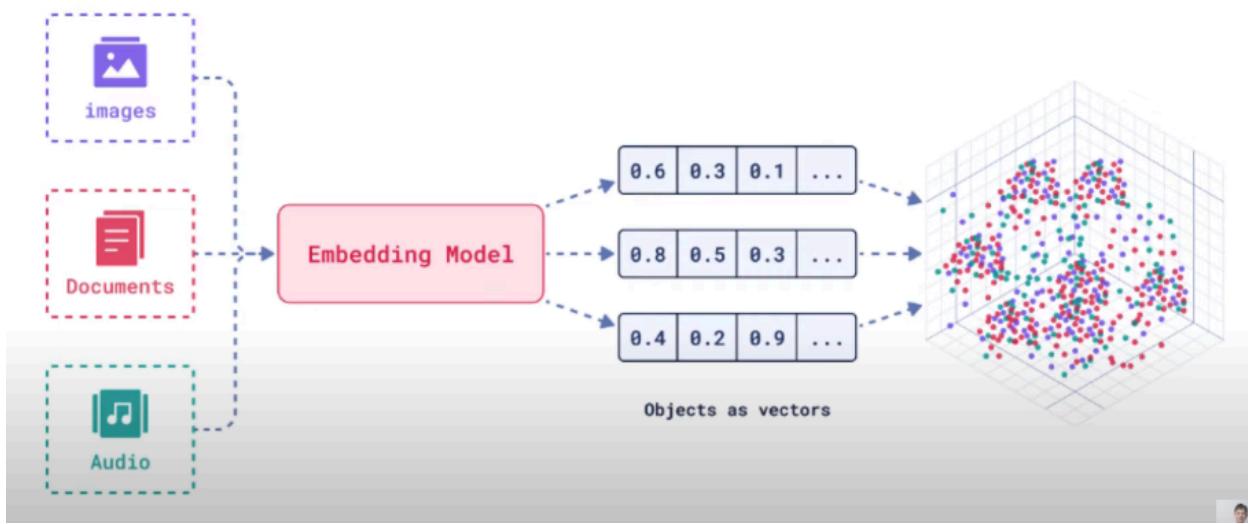
Higher the value of λ ,
higher the penalty

LASSO	Ridge	Elastic Net
- Shrinks coefficients to 0 - Good for variable selection	Makes coefficients smaller	Tradeoff between variable selection and small coefficients
		
$\dots + \lambda \ \theta\ _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda \ \theta\ _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda \left[(1 - \alpha) \ \theta\ _1 + \alpha \ \theta\ _2^2 \right]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

Distance function - hàm đo khoảng cách - đo khoảng cách và độ tương đồng giữa các điểm dữ liệu

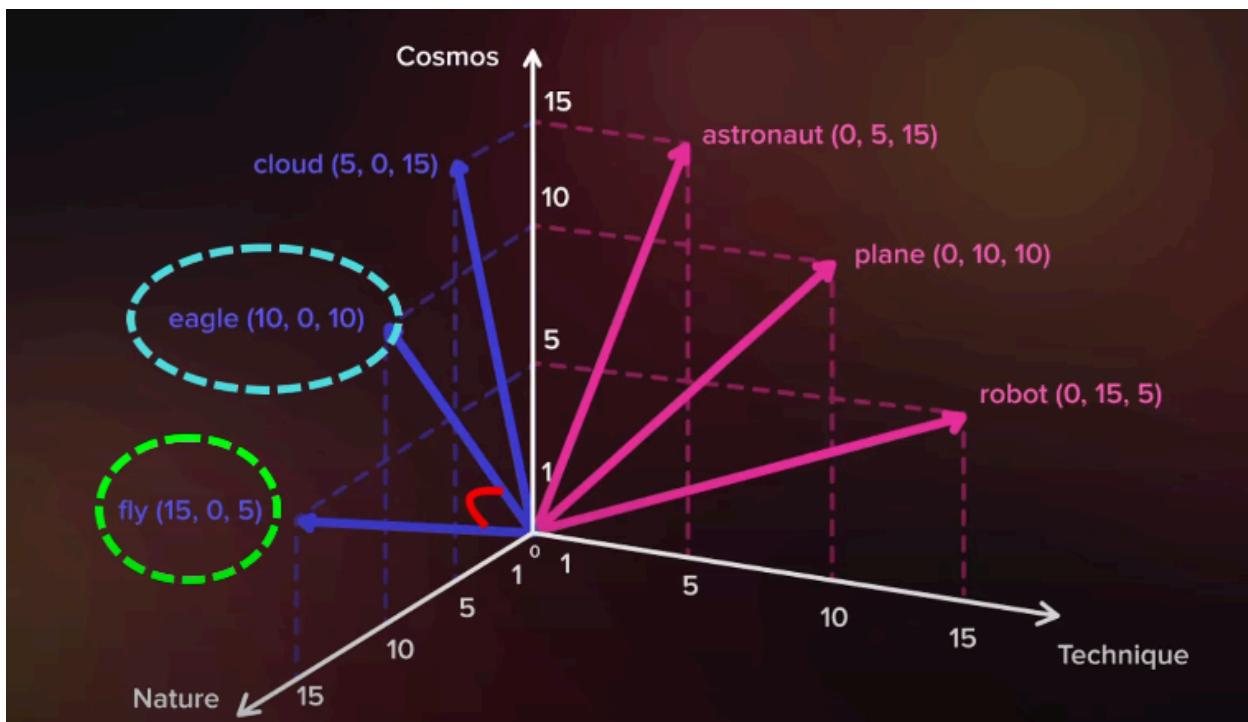


Embedding function - giúp biến đổi các dữ liệu đầu vào sang vector nhiều chiều



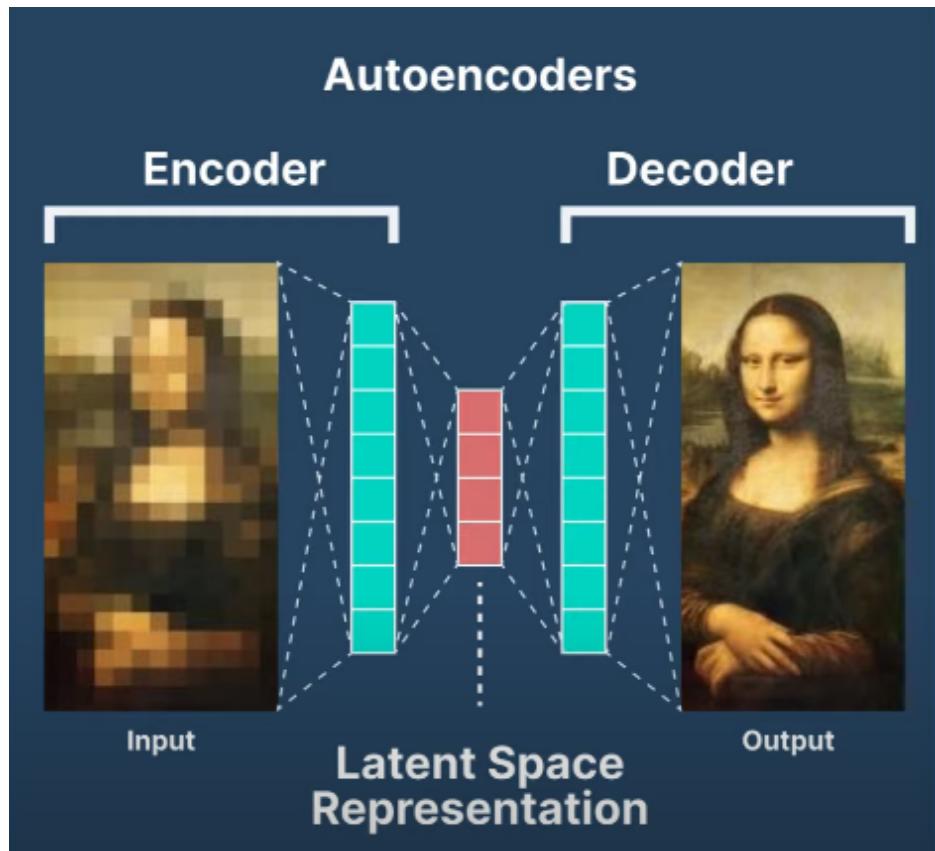
worff2vec and gloVe được sử dụng trong ngôn ngữ tự nhiên và autoencoder được sử dụng trong xử lý hình ảnh anam thân

worff2vec giúp biến từ thành vector số sao cho các từ có ý nghĩ tương tự nhau sẽ có vector gâng nhau trong không gian

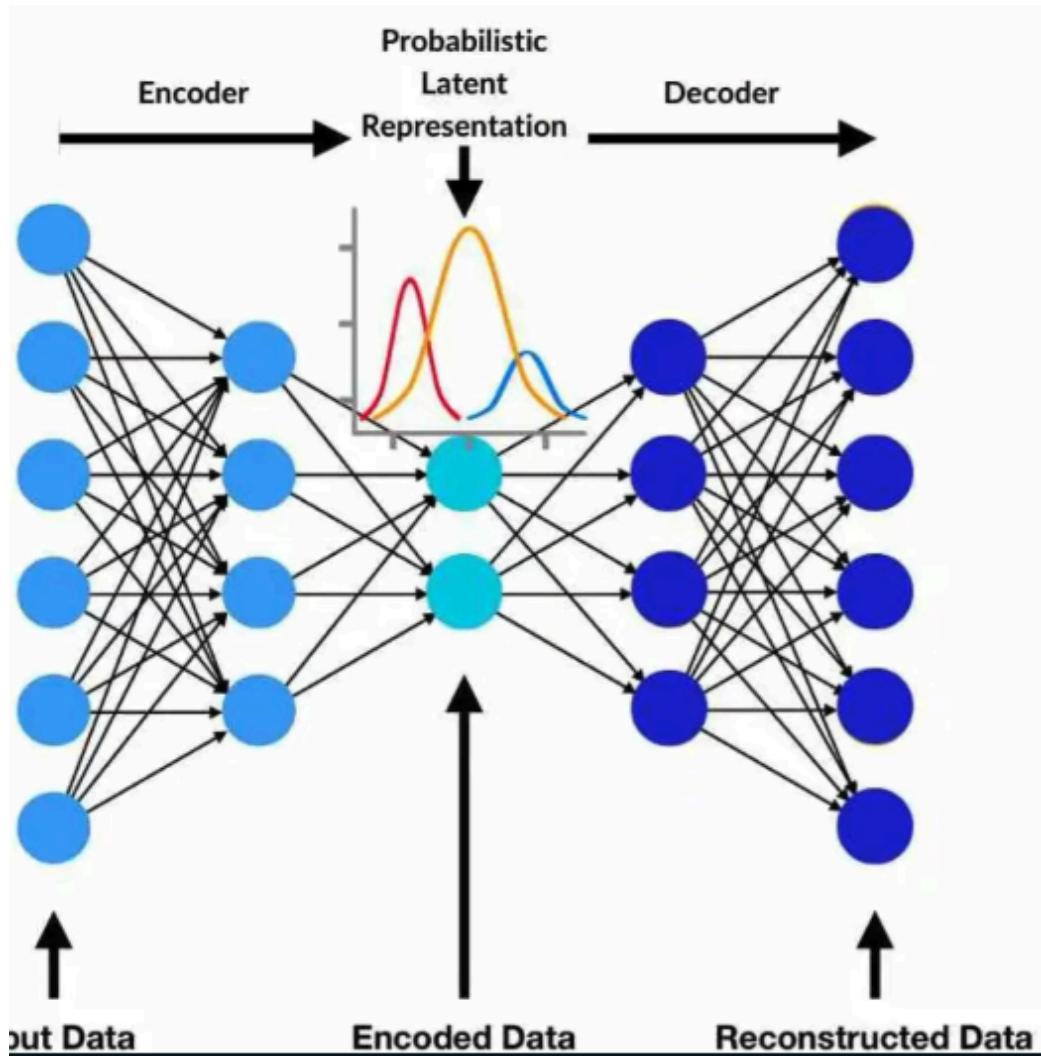


GloVe: để chuyển đổi từ ngữ thành vector dựa trên thống kê xuất hiện đồng thời trong văn bản .

Autorencode cơ bản là 1 mạng nơ rô được dùng để học cách mã hóa và giả mã dữ liệu một cách hiệu quả giúp giảm kích thước hoặc loại bỏ nhiễu của dữ liệu

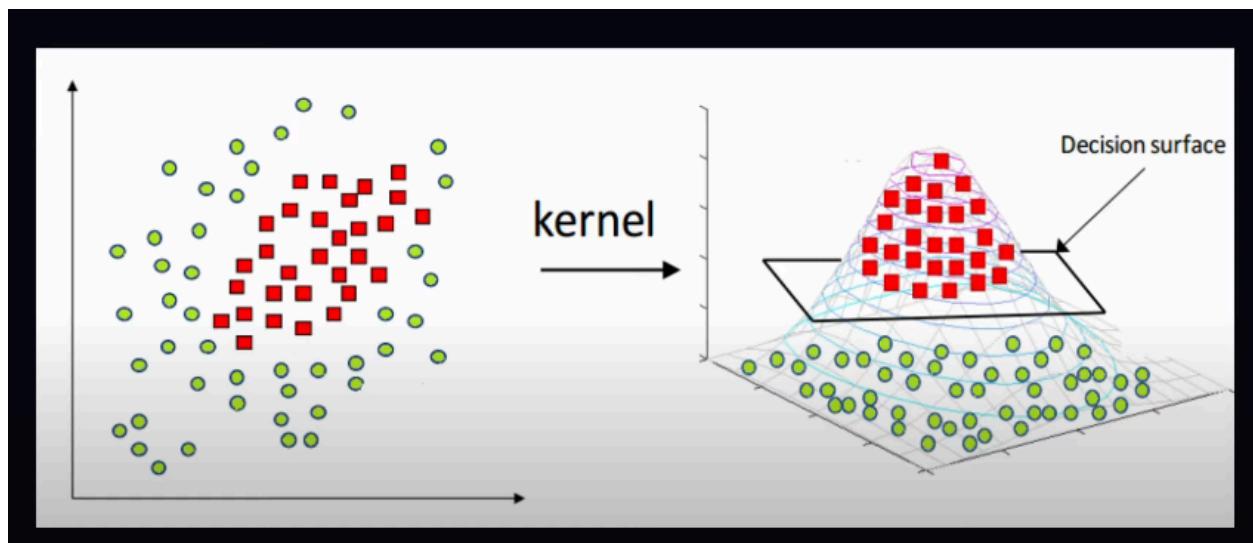


được tọa ra bởi Encoder



Kernel function - được dùng trong SVM(1 thuật toán được coi là quan trọng nhất trong ML)

giúp biến dữ liệu trong không gian ban đầu thành không gian cao hơn để dễ dàng phân类 hơn



Machine learning model