



ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

한-베정보통신기술대학교

Chương 5-1. Phân lớp ảnh (Image Classification)

TS. PHẠM NGUYỄN MINH NHỰT

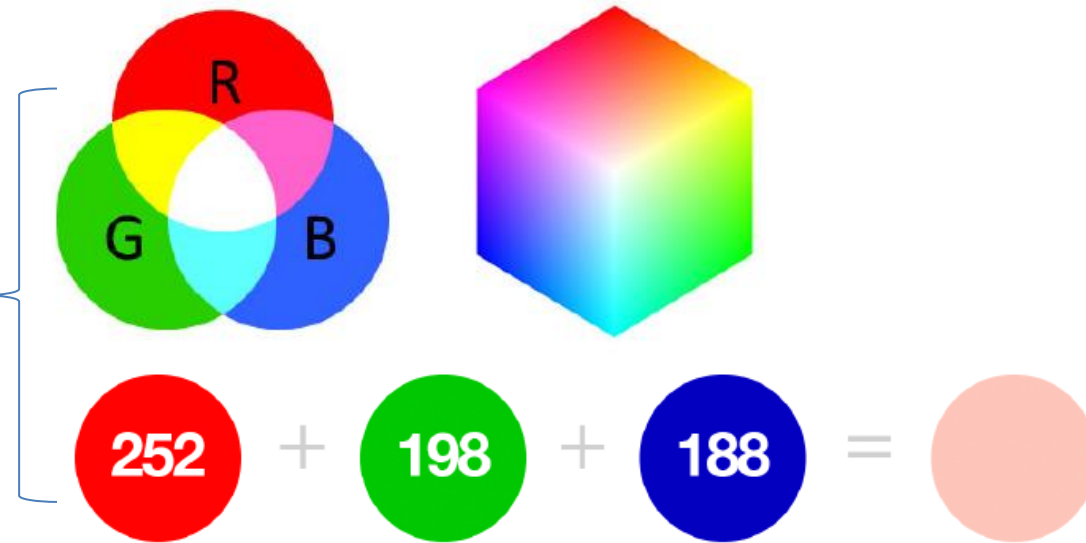
Email: pnmnhut@vku.udn.vn

Phone: 0903.501.421

Ảnh đa cấp xám →



Mô hình màu RGB



Ảnh màu RGB →





Pixel

[30	28	25	...	29	35	29]
[28	29	29	...	27	54	47]
[20	23	27	...	68	55	47]
...							
[123	151	153	...	71	94	94]
[128	120	94	...	88	85	85]
[74	79	79	...	83	65	64]

Minh họa ảnh xám và giá trị mức xám của các điểm ảnh



Pixel

$\begin{bmatrix} 22 & 32 & 21 \end{bmatrix}$
 $\begin{bmatrix} 22 & 32 & 21 \end{bmatrix}$
 $\begin{bmatrix} 22 & 32 & 21 \end{bmatrix}$... $\begin{bmatrix} 15 & 28 & 10 \end{bmatrix}$ $\begin{bmatrix} 35 & 48 & 31 \end{bmatrix}$ $\begin{bmatrix} 32 & 41 & 36 \end{bmatrix}$
 $\begin{bmatrix} 19 & 29 & 18 \end{bmatrix}$ $\begin{bmatrix} 19 & 29 & 18 \end{bmatrix}$ $\begin{bmatrix} 20 & 30 & 19 \end{bmatrix}$... $\begin{bmatrix} 25 & 39 & 14 \end{bmatrix}$ $\begin{bmatrix} 35 & 48 & 28 \end{bmatrix}$ $\begin{bmatrix} 35 & 45 & 34 \end{bmatrix}$
 $\begin{bmatrix} 15 & 25 & 14 \end{bmatrix}$ $\begin{bmatrix} 16 & 26 & 15 \end{bmatrix}$ $\begin{bmatrix} 17 & 27 & 16 \end{bmatrix}$... $\begin{bmatrix} 46 & 62 & 26 \end{bmatrix}$ $\begin{bmatrix} 43 & 58 & 25 \end{bmatrix}$ $\begin{bmatrix} 33 & 47 & 22 \end{bmatrix}$
...
 $\begin{bmatrix} 108 & 140 & 90 \end{bmatrix}$ $\begin{bmatrix} 149 & 179 & 129 \end{bmatrix}$ $\begin{bmatrix} 154 & 180 & 132 \end{bmatrix}$... $\begin{bmatrix} 60 & 76 & 27 \end{bmatrix}$ $\begin{bmatrix} 65 & 85 & 32 \end{bmatrix}$ $\begin{bmatrix} 66 & 91 & 33 \end{bmatrix}$
 $\begin{bmatrix} 106 & 132 & 87 \end{bmatrix}$ $\begin{bmatrix} 116 & 142 & 97 \end{bmatrix}$ $\begin{bmatrix} 99 & 127 & 78 \end{bmatrix}$... $\begin{bmatrix} 87 & 100 & 57 \end{bmatrix}$ $\begin{bmatrix} 84 & 102 & 54 \end{bmatrix}$ $\begin{bmatrix} 79 & 99 & 46 \end{bmatrix}$
 $\begin{bmatrix} 77 & 87 & 62 \end{bmatrix}$ $\begin{bmatrix} 66 & 81 & 48 \end{bmatrix}$ $\begin{bmatrix} 63 & 90 & 45 \end{bmatrix}$... $\begin{bmatrix} 76 & 85 & 54 \end{bmatrix}$ $\begin{bmatrix} 66 & 76 & 41 \end{bmatrix}$ $\begin{bmatrix} 54 & 66 & 26 \end{bmatrix}$

Minh họa ảnh màu RGB và giá trị mức xám của các điểm ảnh

Để chuyển ảnh RGB về ảnh xám $\rightarrow W = (R \times 0.299) + (G \times 0.587) + (B \times 0.114)$

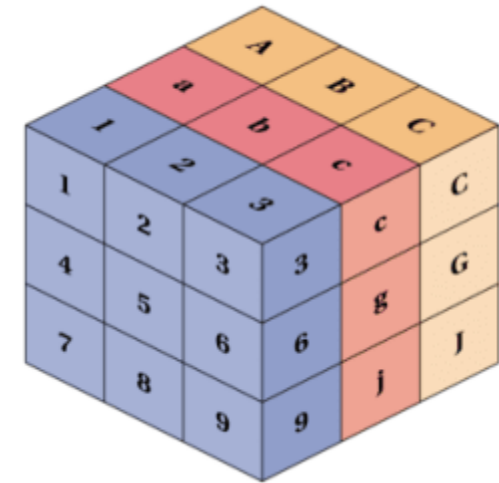
- Dữ liệu biểu diễn 1 chiều → Gọi là Vektor → Thông thường biểu diễn dưới dạng cột → Kích thước N
- Dữ liệu biểu diễn 2 chiều → Gọi là Matrix → Kích thước M x N
- Dữ liệu biểu diễn hơn 2 chiều → Gọi là Tensor → Nếu là K chiều thì kích thước M x N x K

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Vector

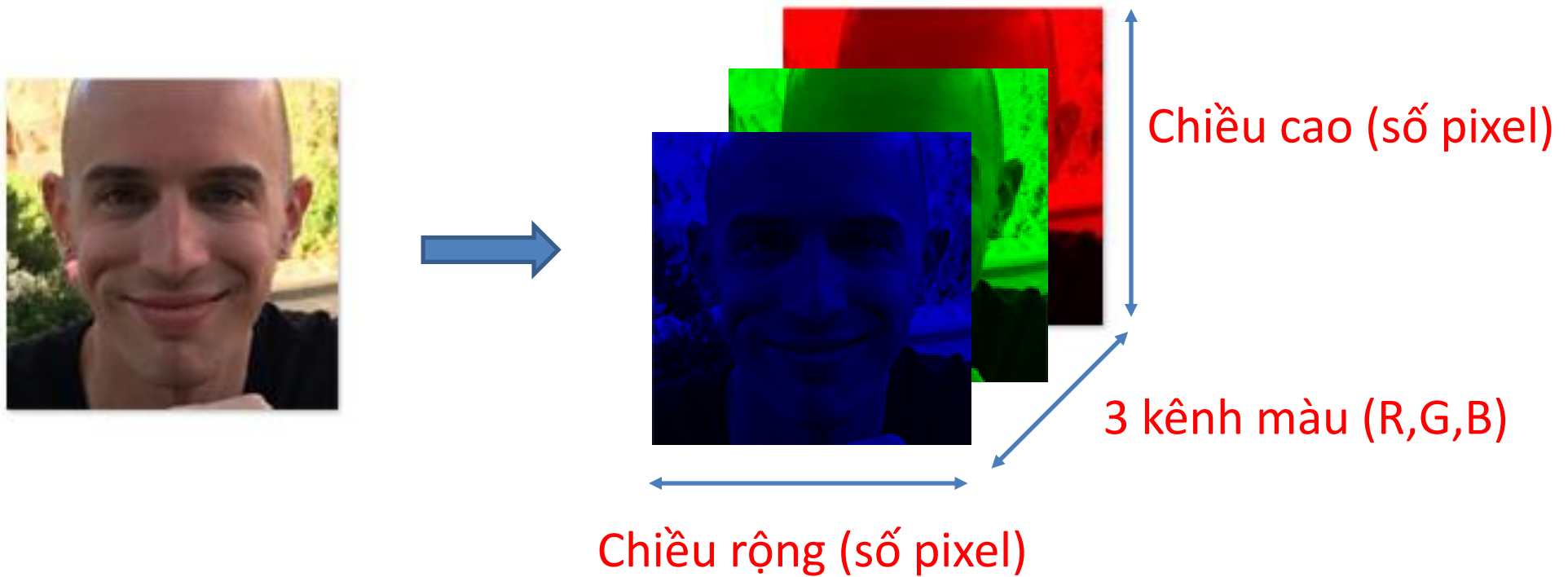
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Matrix



Tensor 3D

- Ảnh màu → Tensor 3 chiều



- Video → Tensor 4 chiều



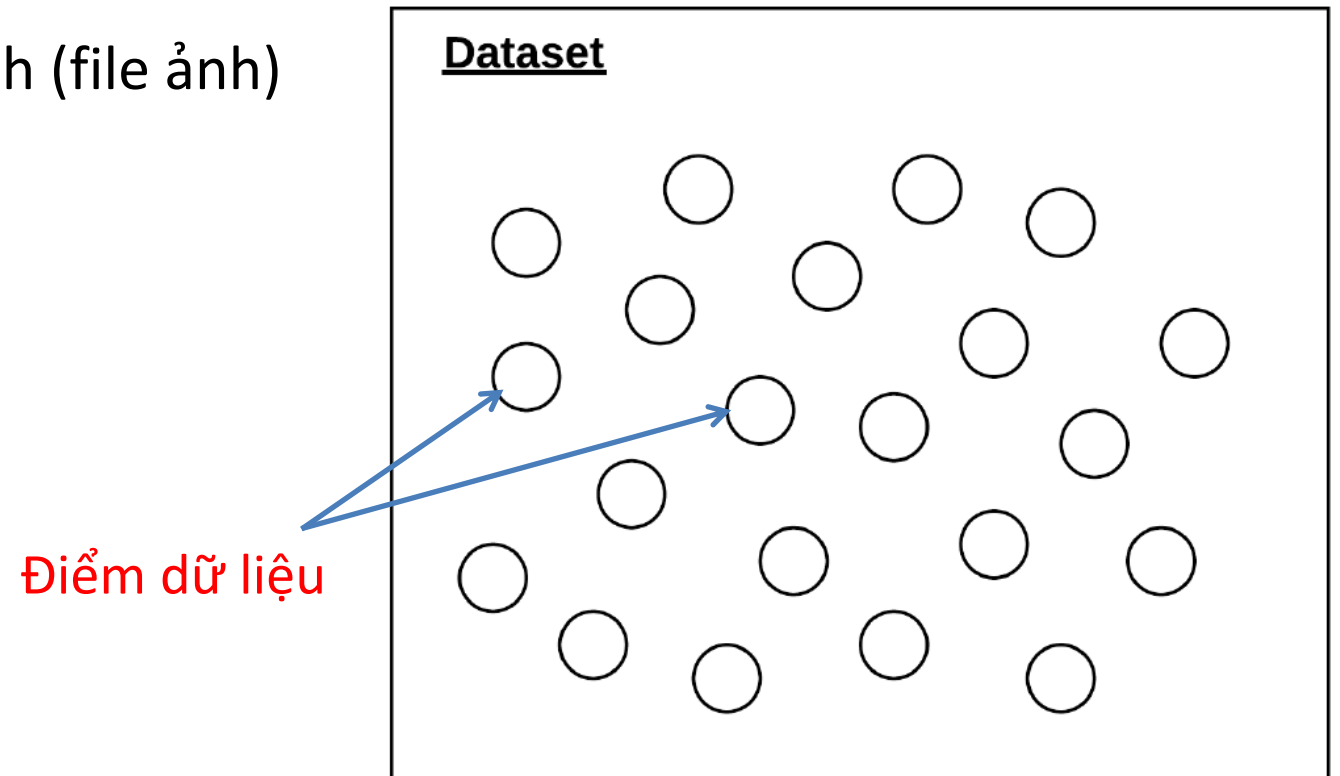
- Xét một bức ảnh mô tả cảnh một bờ biển
- Mô tả các thuộc tính:
 - Không gian: Bầu trời ở trên cùng và cát/đại dương ở dưới cùng
 - Màu sắc: Bầu trời xanh đậm, nước biển xanh nhạt hơn bầu trời, trong khi cát có màu râm nắng.
 - Kết cấu: Bầu trời có hoa văn tương đối đồng nhất, trong khi cát không đồng nhất



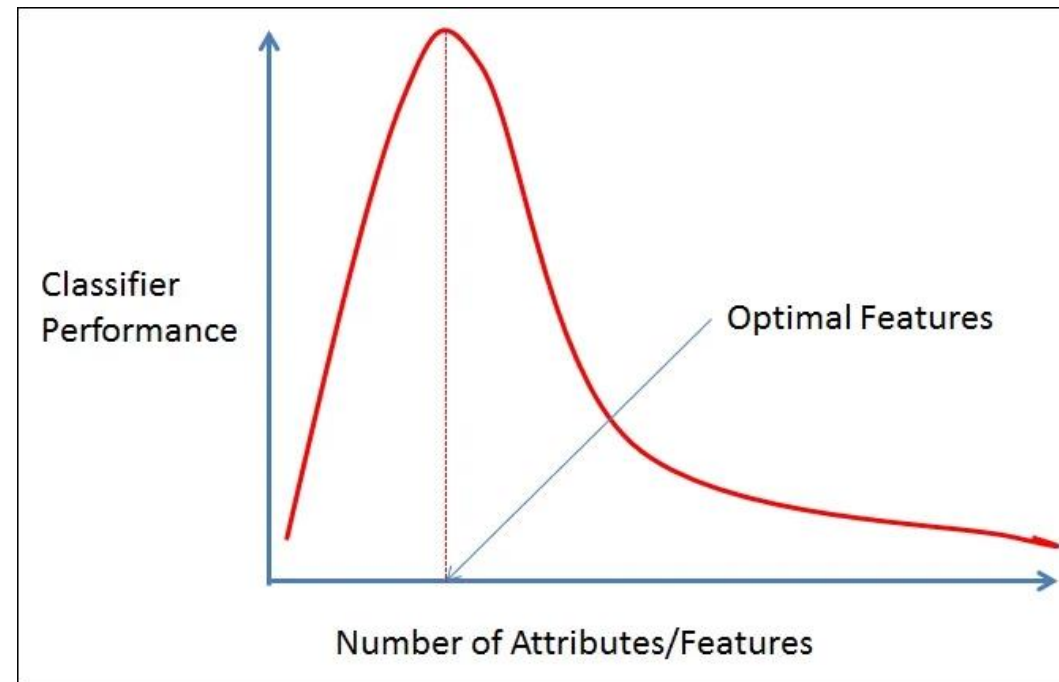
➔ Các thuộc tính này gọi là các đặc trưng

➔ Để mã hóa các đặc trưng này ➔ rút trích các đặc trưng (Chương 4)

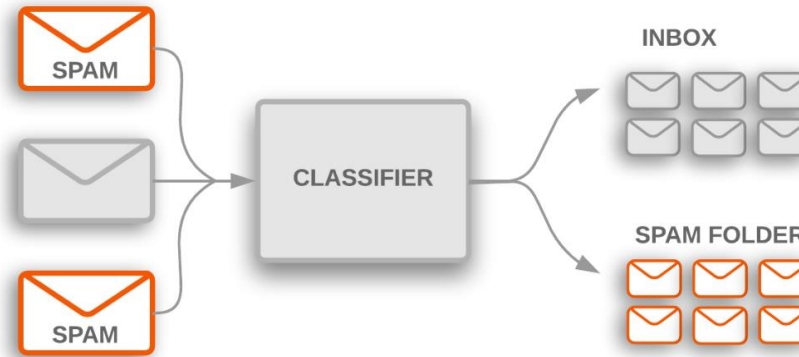
- **Dataset** (Tập dữ liệu): sử dụng để trích xuất kiến thức từ nó (dạy cho máy học)
- **Data point** (Điểm dữ liệu): mỗi mục dữ liệu trong tập dữ liệu gọi là một điểm dữ liệu.
- Đối với bài toán phân loại ảnh:
 - Tập dữ liệu là một tập hợp các ảnh (file ảnh)
 - Mỗi file ảnh là một điểm dữ liệu



- **Kích thước/chiều dữ liệu (Dimensionality):**
 - Kích thước dữ liệu là số thuộc tính hay số đặc trưng của đối tượng trong tập dữ liệu
 - Trong một tập dữ liệu cụ thể, phân tích dữ liệu sẽ khó khăn khi kích thước tăng
 - **Đối với bài toán phân loại**, nếu không đủ đối tượng dữ liệu thì việc tạo mô hình sẽ không tin cậy (cho tất cả các đối tượng)



- Bài toán phân loại nói chung** là quá trình phân chia dữ liệu thành các danh mục hoặc lớp riêng biệt



- Phân loại/lớp ảnh (image Classification)** là gán nhãn cho một ảnh từ một bộ danh mục ảnh → Phân tích một ảnh đầu vào và trả về một nhãn cho ảnh (Nhãn từ một tập các danh mục được xác định trước).



Từ ảnh đầu vào, nhiệm vụ là phân lớp ảnh vào nhóm con vật nào → Gắn nhãn cho nó

- **Ví dụ:**

- Giả sử có tập các danh mục: categories = {cat, dog, panda} và một ảnh như hình
- Mục tiêu là lấy ảnh đầu vào này và gán nhãn cho nó từ bộ danh mục
- Hệ thống phân loại gán nhãn cho ảnh với xác suất bất kỳ: dog (95%); cat (4%); panda (1%).

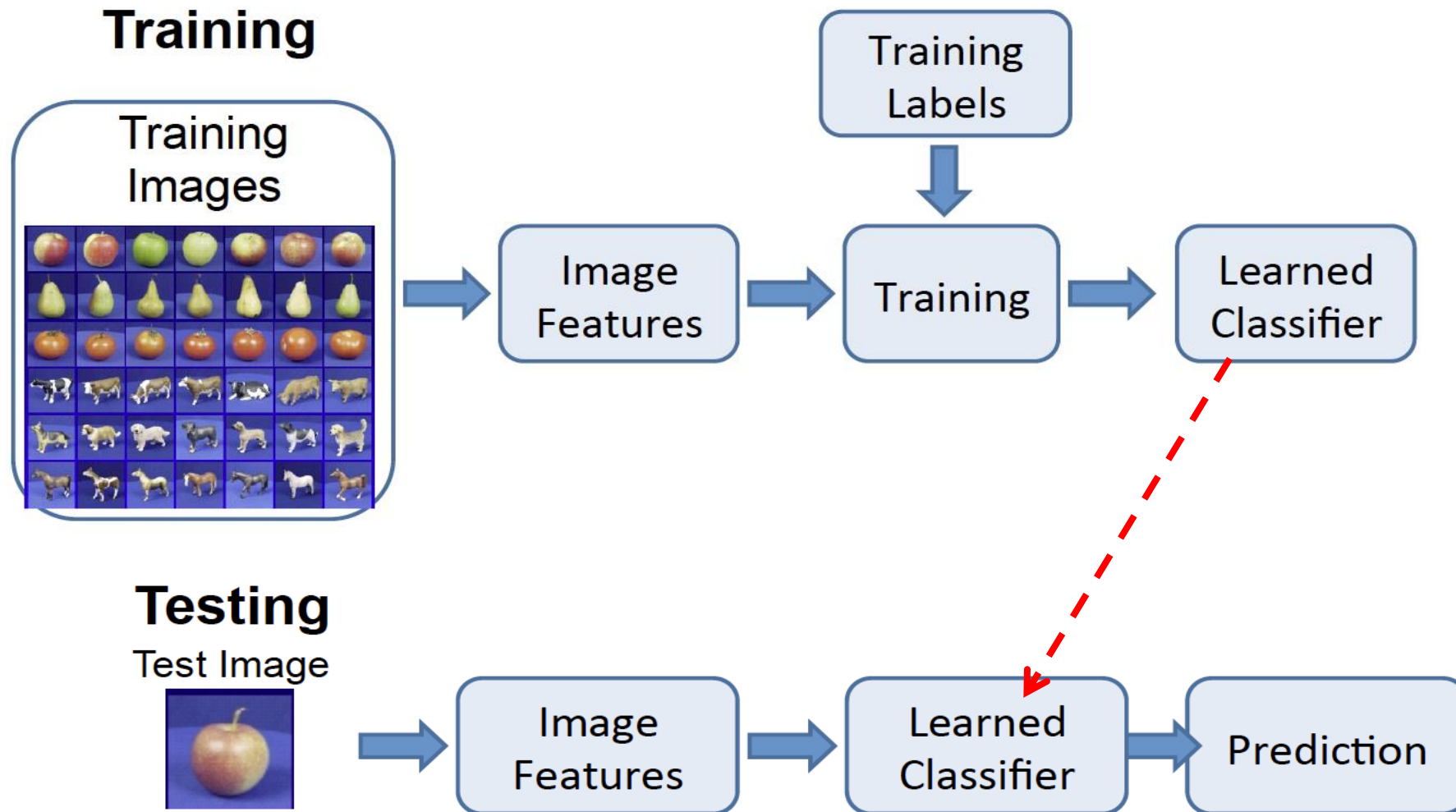


- **Tổng quát bài toán phân lớp ảnh:**

- Với ảnh đầu vào có kích thước $W \times H$
 - Ảnh có 3 kênh: Red, Green và Blue
- ➔ Mục tiêu: với ảnh có kích thước $W \times H \times 3$ pixel và tìm cách phân loại chính xác ảnh (dự đoán càng chính xác càng tốt).

- **Học máy** (Machine Learning - ML) là nghiên cứu về các thuật toán và mô hình toán học (thường mô hình thống kê) cho phép các chương trình máy tính tự động cải thiện (nên gọi là quá trình học) thông qua kinh nghiệm.
- ➔ Đó là khoa học làm cho máy tính hoạt động bằng cách cung cấp dữ liệu cho chúng và để chúng tự học.
- **Các loại thuật toán học:**
 - Học có giám sát - Supervised Learning
 - Học không giám sát - Unsupervised Learning
 - Học bán giám sát - Semi Supervised Learning

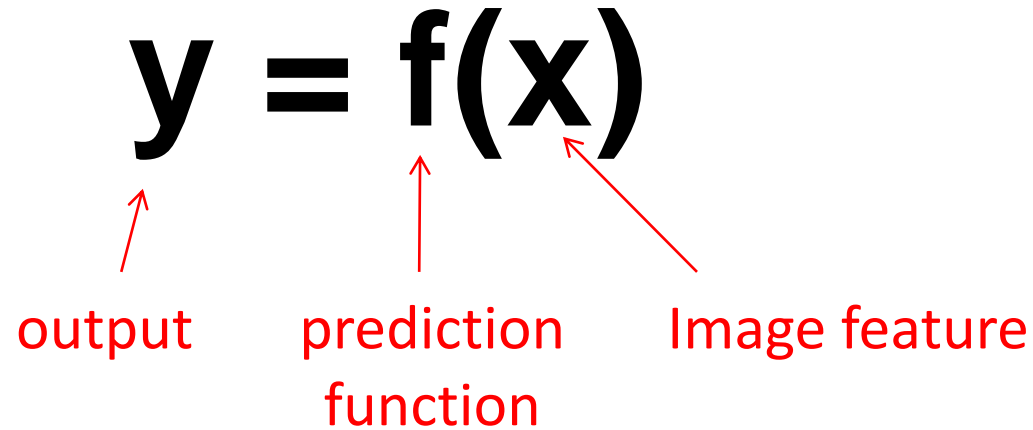
- Các giai đoạn của kỹ thuật Học máy



- Hàm dự đoán

$$y = f(x)$$

output prediction function Image feature



- Training: Cho một tập mẫu dữ liệu có N phần tử đã được gán nhãn $\{(x_1, y_1), \dots, (x_N, y_N)\}$, áp dụng hàm f để dự đoán nhãn kết quả đầu ra y sao cho tối thiểu lỗi dự đoán trên tập huấn luyện
- Testing: áp dụng hàm dự đoán f cho mẫu dữ liệu test x (dữ liệu này chưa được dùng để training) và đầu ra là giá trị dự đoán $y = f(x)$

- Ví dụ học có giám sát đối với phân loại ảnh (02 loại/lớp):
 - Giả sử cho tập dữ liệu ảnh cùng với nhãn tương ứng để dạy máy học

Label	R_μ	G_μ	B_μ	R_σ	G_σ	B_σ
Cat	57.61	41.36	123.44	158.33	149.86	93.33
Cat	120.23	121.59	181.43	145.58	69.13	116.91
Cat	124.15	193.35	65.77	23.63	193.74	162.70
Dog	100.28	163.82	104.81	19.62	117.07	21.11
Dog	177.43	22.31	149.49	197.41	18.99	187.78
Dog	149.73	87.17	187.97	50.27	87.15	36.65

- Cột đầu tiên là nhãn, 6 cột còn lại tương ứng với vector đặc trưng: giá trị trung bình và độ lệch chuẩn cho mỗi kênh màu RGB
- Thuật toán học có giám sát sẽ đưa ra dự đoán trên từng vector đặc trưng → Nếu nó đưa ra dự đoán không chính xác → sửa nó bằng cách cho nó biết nhãn chính xác là gì → Quá trình này tiếp tục cho đến khi đáp ứng tiêu chí dừng mong muốn, chẳng hạn như độ chính xác, số lần lặp của quá trình học tập hoặc thời gian...

- Ví dụ học không sát đối với phân loại ảnh (02 loai/lớp):

- Học không giám sát (đôi khi được gọi là học tự học) → loại bỏ cột nhãn → không có nhãn nào được liên kết với dữ liệu đầu vào → không thể sửa mô hình nếu nó dự đoán sai

R_μ	G_μ	B_μ	R_σ	G_σ	B_σ
57.61	41.36	123.44	158.33	149.86	93.33
120.23	121.59	181.43	145.58	69.13	116.91
124.15	193.35	65.77	23.63	193.74	162.70
100.28	163.82	104.81	19.62	117.07	21.11
177.43	22.31	149.49	197.41	18.99	187.78
149.73	87.17	187.97	50.27	87.15	36.65

- Để các thuật toán học không giám sát thành công → tìm hiểu cấu trúc của tập dữ liệu và áp dụng các tính năng đã học của học có giám sát
- Với trường hợp có quá ít dữ liệu → Thường dùng các thuật toán học không giám sát: Principle Component Analysis (PCA), phân cụm k-means

- Ví dụ học bán giám sát đối với phân loại ảnh (02 loại/lớp):
 - Chỉ có một số nhãn được liên kết với dữ liệu → sử dụng học bán giám sát

Label	R_μ	G_μ	B_μ	R_σ	G_σ	B_σ
Cat	57.61	41.36	123.44	158.33	149.86	93.33
?	120.23	121.59	181.43	145.58	69.13	116.91
?	124.15	193.35	65.77	23.63	193.74	162.70
Dog	100.28	163.82	104.81	19.62	117.07	21.11
?	177.43	22.31	149.49	197.41	18.99	187.78
Dog	149.73	87.17	187.97	50.27	87.15	36.65

- Thuật toán học bán giám sát sẽ lấy các phần dữ liệu đã biết → phân tích và gán nhãn cho những điểm dữ liệu chưa được gán nhãn để làm dữ liệu huấn luyện bổ sung
- Quá trình này có thể lặp lại khi thuật toán bán giám sát học “cấu trúc” của dữ liệu để đưa ra dự đoán chính xác hơn và tạo dữ liệu đào tạo đáng tin cậy hơn.

- Underfitting:

- Một thuật toán ML được cho là Underfitting khi nó không thể nắm bắt được xu hướng cơ bản của dữ liệu → nghĩa là mô hình không đủ “phù hợp” với dữ liệu.
- Underfitting thường xảy ra khi có ít dữ liệu để huấn luyện mô hình

- Overfitting:

- Khi một mô hình được đào tạo với quá nhiều dữ liệu → nó học từ luôn từ dữ liệu không chính xác và nhiễu
- → Sử dụng tính năng xác thực chéo để giảm tình trạng trang bị thừa dữ liệu
- → Điều chỉnh siêu tham số với tập huấn luyện ban đầu
- → Dùng bộ dữ liệu test chưa từng dùng (train) để kiểm tra/chọn mô hình cuối cùng

- **Accuracy:**

- Tính toán độ chính xác dự đoán của mô hình
- ➔ cho biết trong số tất cả các điểm dữ liệu thì có bao nhiêu phần trăm điểm dữ liệu được dự đoán đúng
- Bài toán phân loại ảnh ➔ điểm dữ liệu là file ảnh đầu vào

$$\text{Accuracy} = \frac{\text{\# of correct Prediction}}{\text{Total \# of points}} \times 100$$

- **Confusion Metrics:** sử dụng ma trận 2x2

- Hàng: Thực tế (Actual)
- Cột: Dự đoán (Predicted)

Confusion Matrix		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

- **Ví dụ: Xét bài toán phân lớp (2 lớp):** Positive và Negative

- Nếu điểm dữ liệu thực tế là Positive và được dự đoán là Positive → thì nhận được True Positive
 - True: có nghĩa là dự đoán phân loại chính xác
- Nếu điểm dữ liệu thực tế là Negative và được dự đoán là Positive → tức là dự đoán không chính xác → thì nhận được False Positive
 - False: có nghĩa là dự đoán phân loại không chính xác

- **Precision:** là thước đo cho biết, trong số tất cả các lớp Positive được dự đoán, có bao nhiêu lớp thực sự là Positive, tức là:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- **Recall:** là thước đo cho biết, trong số tất cả các lớp Positive, có bao nhiêu lớp thực sự được dự đoán chính xác, tức là:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- **F1 - Score**

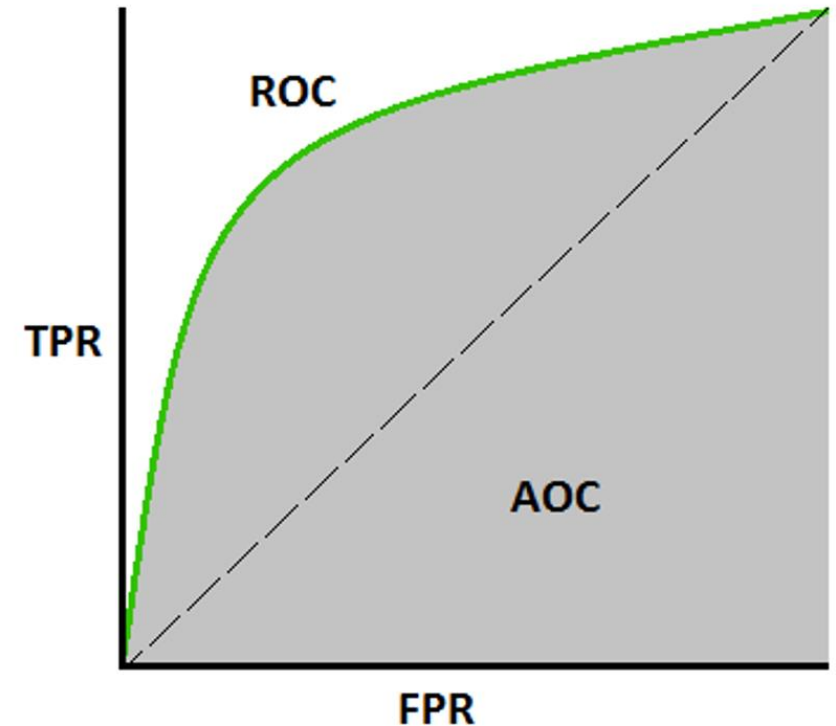
- Là giá trị trung bình Harmonic của Precision và Recall, trong đó F1 - Score đạt giá trị tốt nhất ở mức 1. F1 - Score còn được gọi là hệ số Sorensen Dice Coefficient (DSC):

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Một mô hình lý tưởng sẽ có Precision và Recall là 1, lúc đó F1 - Score cũng sẽ là 1

• AU-ROC

- Là vùng dưới đường cong Receiver Operating Curve – ROC. Trong đó trục X là FPR và trục Y là TPR
- $TPR = \text{True Positive} / (\text{True Positive} + \text{False Negative})$
- $FPR = \text{False Positive} / (\text{False Positive} + \text{True Negative})$
- ➔ Hiển thị hiệu suất của mô hình theo giá trị ngưỡng.
- ➔ Biểu diễn là mối quan hệ giữa TPR và FPR ở các ngưỡng phân loại khác nhau.
- ➔ Hạ thấp ngưỡng: sẽ phân loại nhiều mục Positive hơn, nhưng làm tăng cả False Positive và True Positives.
- ➔ Vùng dưới đường cong ROC là hiệu suất thực tế của mô hình ở các ngưỡng khác nhau



- **Phương pháp Học máy**
 - Thuật toán Nearest Neighbor
 - Thuật toán kNN (“k-Nearest Neighbors”)
 - Thuật toán Support Vector Machine
 - Phương pháp Neural Network
 - ...
- **Phương pháp học sâu**
 - Phương pháp CNN

- Thuật toán Nearest Neighbor (NN)

- Train (huấn luyện/đào tạo) → model
 - Nhớ tất cả các ảnh đã được train và nhãn của nó
- Dự đoán → nhãn của ảnh cần dự đoán
 - Tìm ảnh đã train gần nhất (giống nhất): thường dùng phép đo khoảng cách từng cặp ảnh giữa ảnh test và mỗi ảnh trong tập dữ liệu
 - Dự đoán nhãn của nó (càng giống nhãn thật càng tốt)
 - Ví dụ:

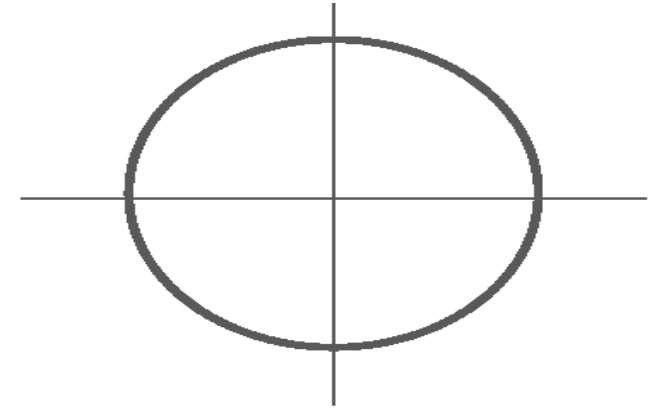
```
def train(train_images, train_labels):
    # build a model for images -> labels...
    return model

def predict(model, test_images):
    # predict test_labels using the model...
    return test_labels
```

- **Thước đo khoảng cách**

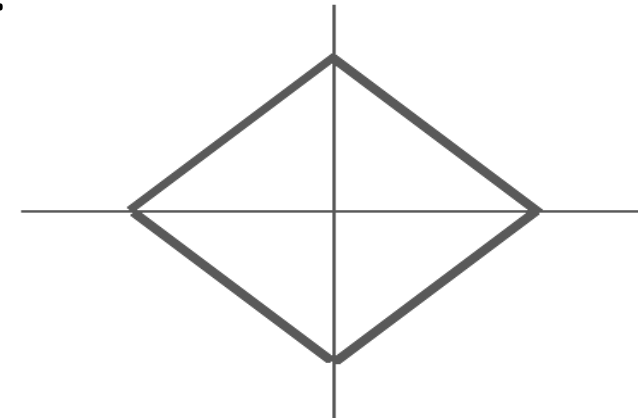
- Khoảng cách Euclide (thường được gọi là khoảng cách L2):

$$d(p, q) = \sqrt{\sum_{i=1}^N (q_i - p_i)^2}$$



- Khoảng cách Manhattan/city block (thường gọi là khoảng cách L1):

$$d(p, q) = \sum_{i=1}^N |q_i - p_i|$$



- Trong đó q_i và p_i là điểm dữ liệu (Data Point) thứ i (là ảnh thứ i)

- Ví dụ: Tính L1

$$d(p, q) = \sum_{i=1}^N |q_i - p_i|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

- Ví dụ: CIFAR-10 (10 nhãn, 50,000 ảnh train; 10,000 ảnh test)



Ảnh Test

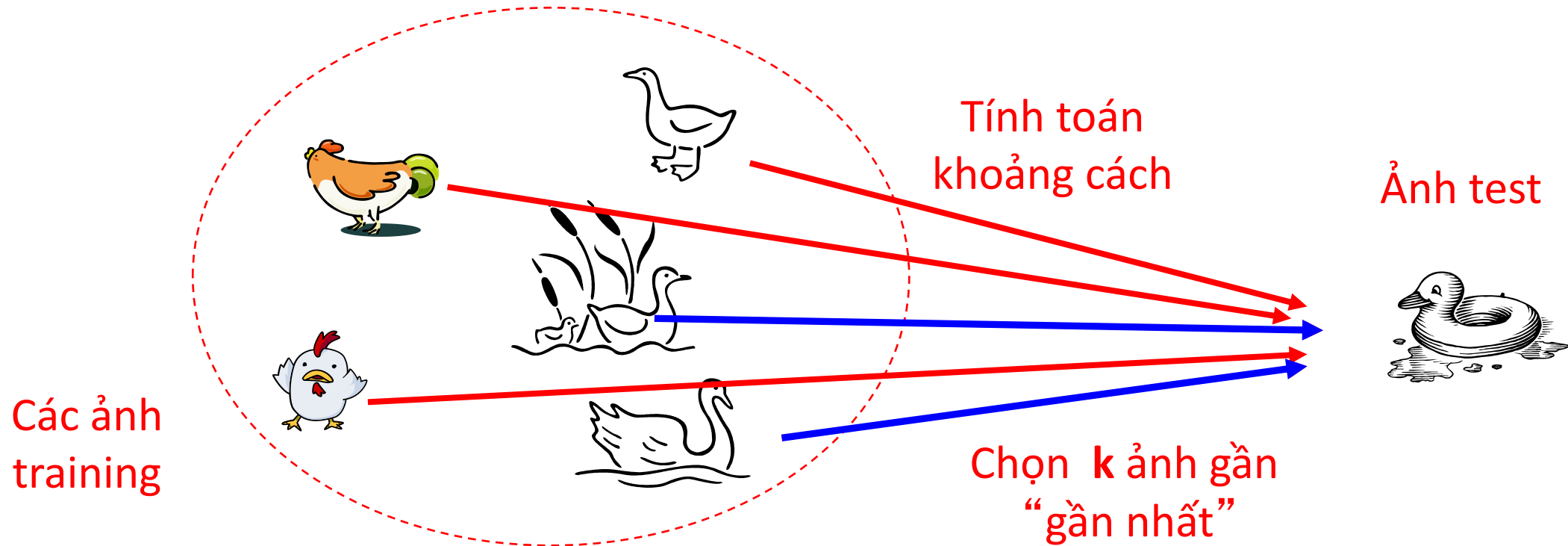


Ảnh kết quả



- Thuật toán k - Nearest Neighbor (k-NN)

- Trường hợp mở rộng của thuật toán NN → tức là xét k ảnh “gần nhất”

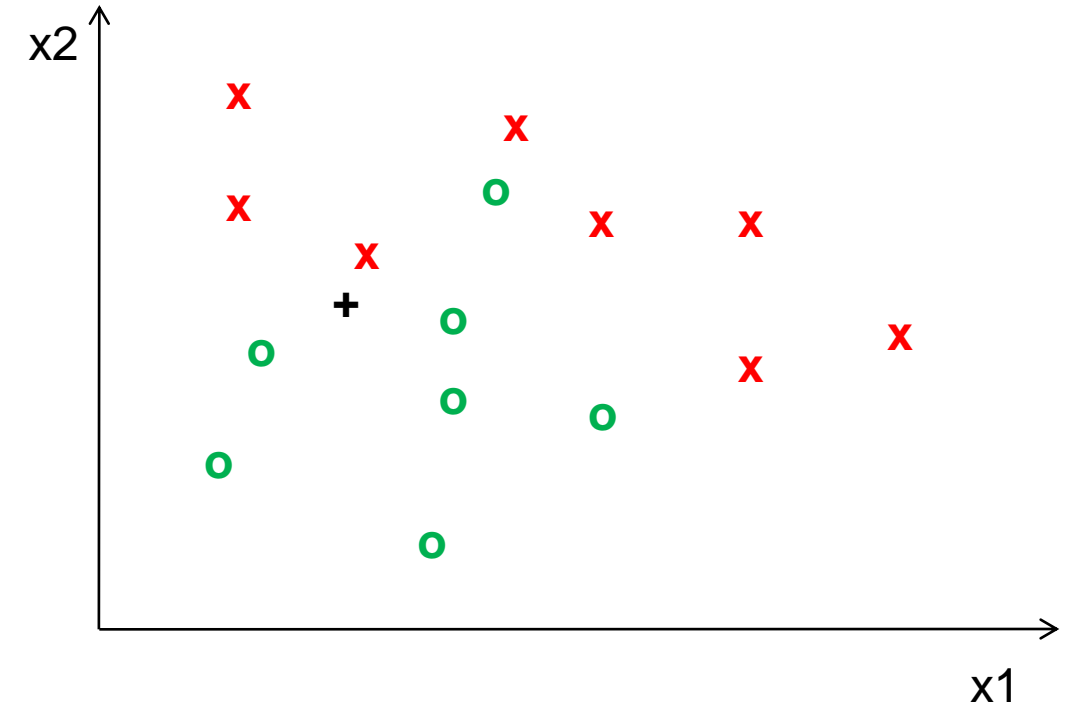


4. Một số phương pháp phân loại ảnh – k-Nearest Neighbor (NN)

- Đo khoảng cách - Euclidean

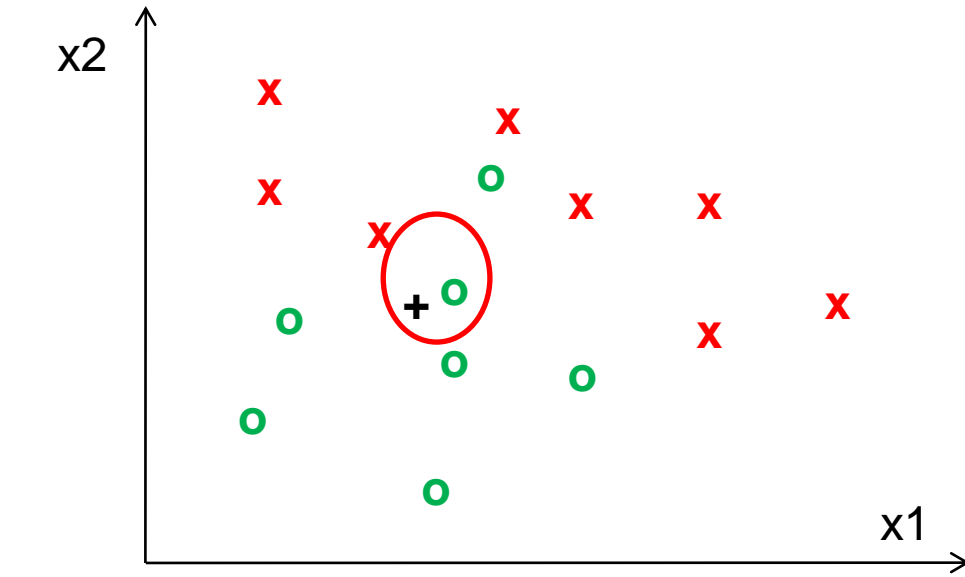
$$d(p, q) = \sqrt{\sum_{i=1}^N (q_i - p_i)^2}$$

$$d(p, q) = \sum_{i=1}^N |q_i - p_i|$$

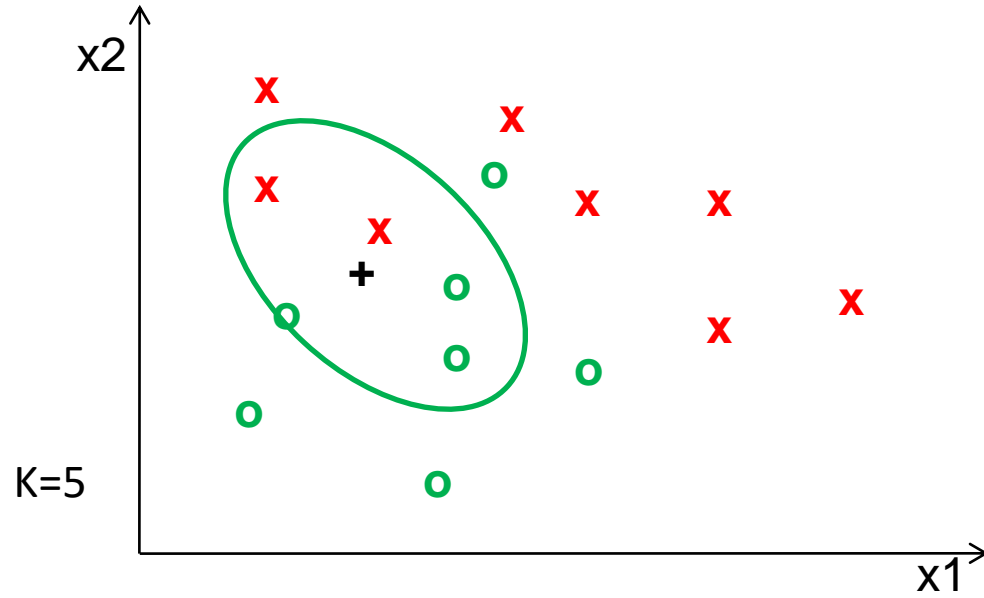


- Trong đó q_i và p_i là điểm dữ liệu (Data Point) thứ i
- x_1, x_2 : đặc trưng của điểm dữ liệu

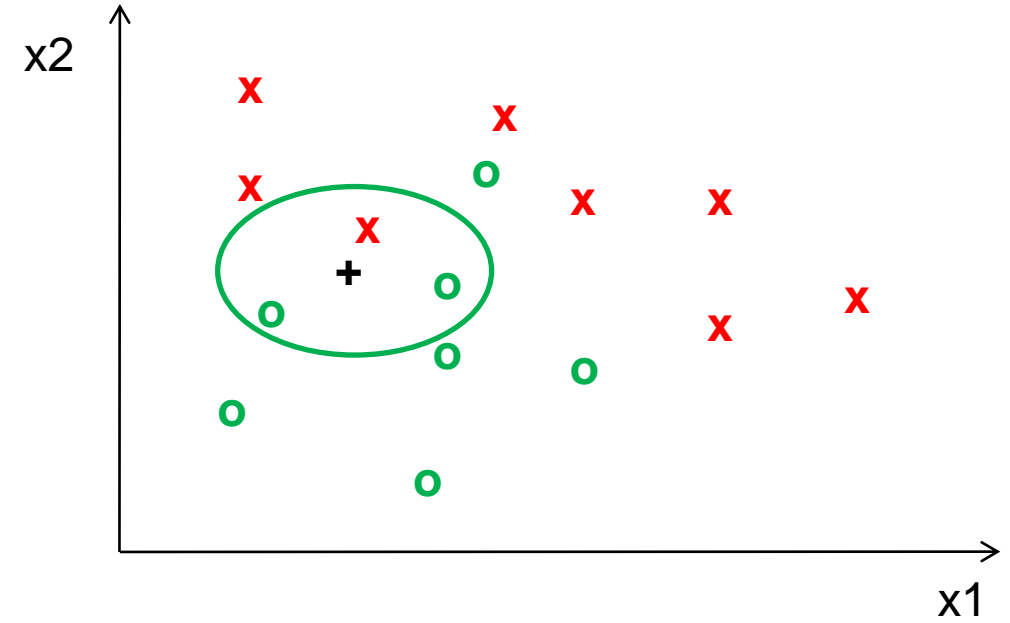
4. Một số phương pháp phân loại ảnh – k-Nearest Neighbor (NN)



K=1

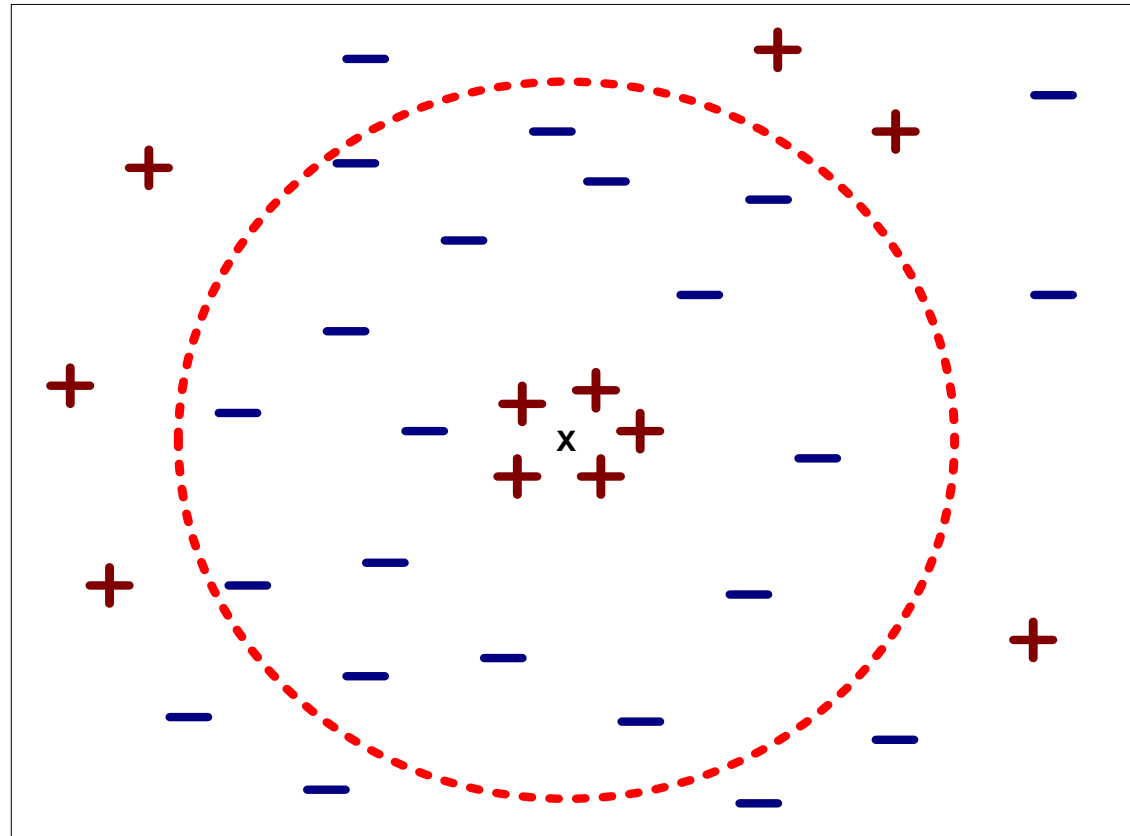


K=5

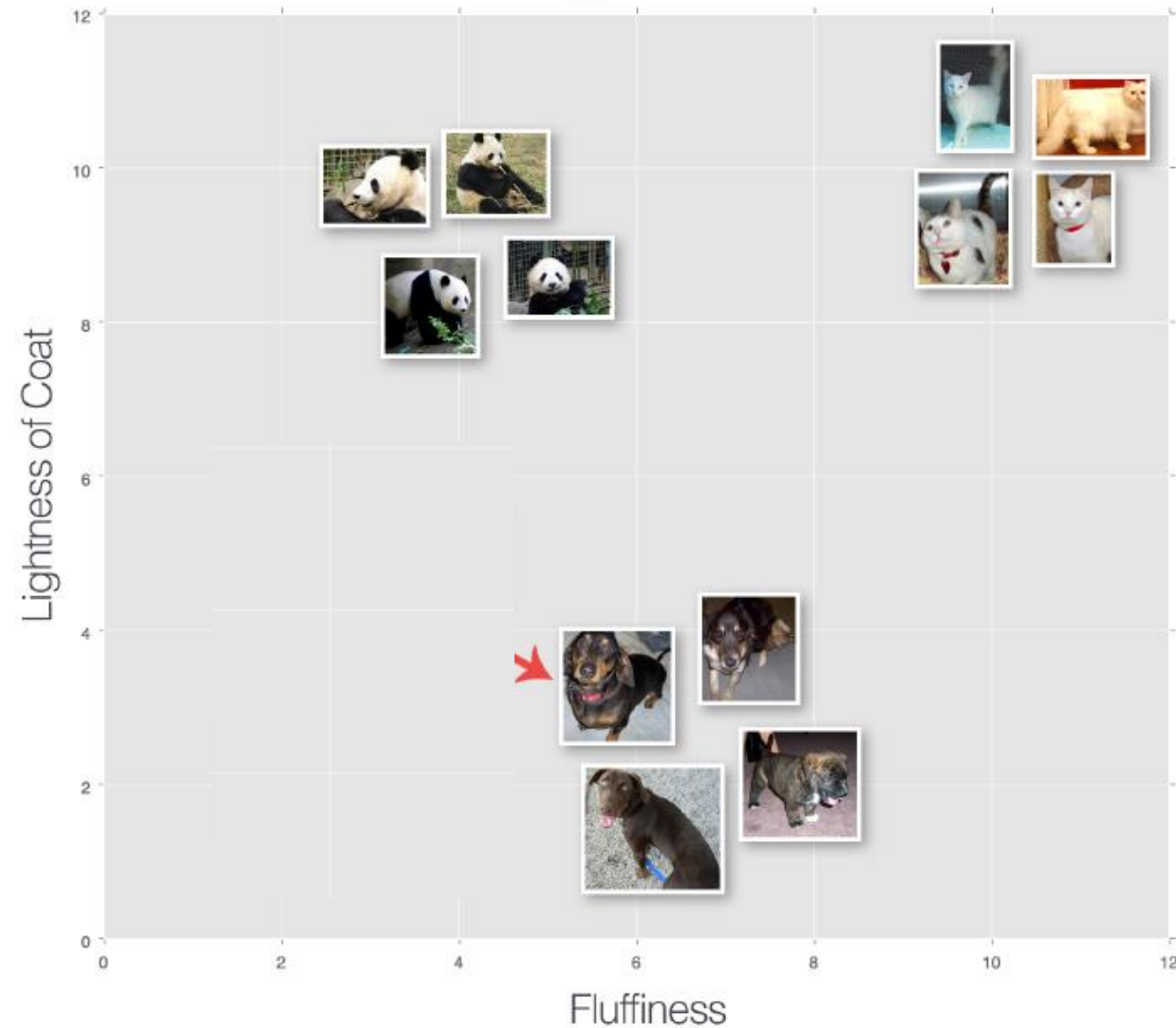


K=3

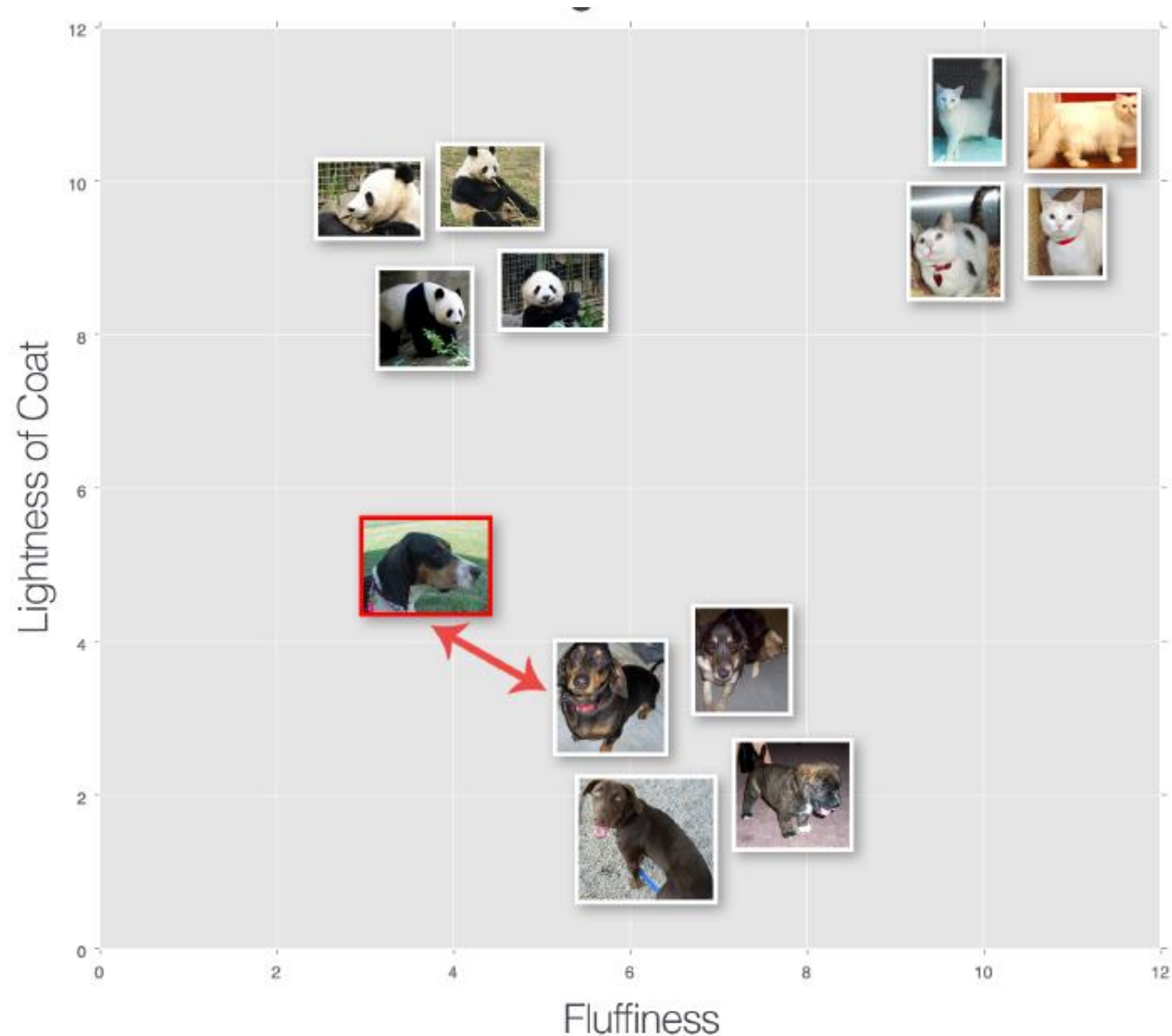
- Chọn giá trị k :
 - Nếu quá nhỏ, nhạy cảm với các nhiễu
 - Nếu quá lớn, khu vực lân cận có thể bao gồm các điểm dữ liệu của các lớp khác



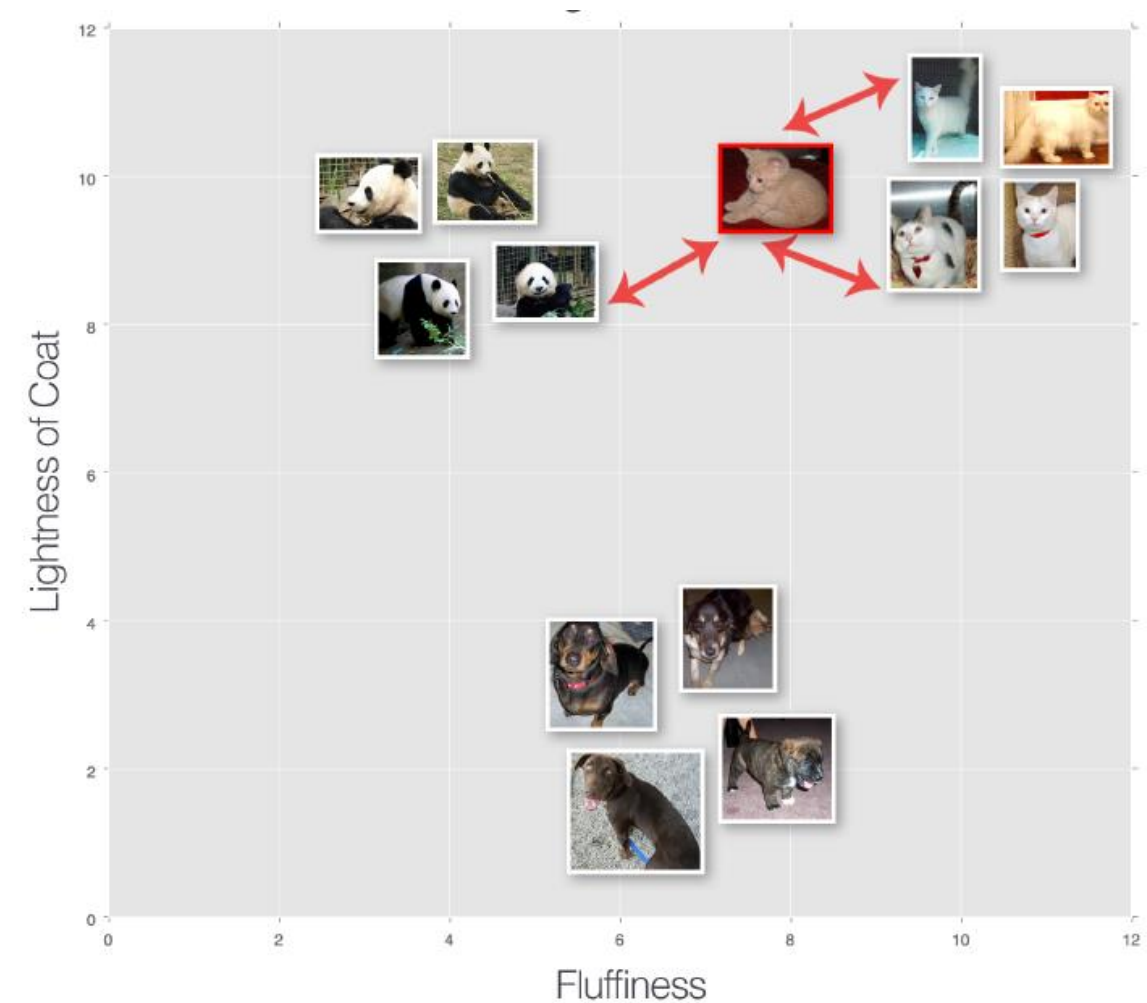
- Ví dụ: Xét bộ phân lớp (03 lớp), thuật toán k-NN hoạt động như sau:
 - Giả sử có Dataset có 03 loại động vật: dog, cat và panda
 - Thuật toán dựa vào khoảng cách giữa các vectơ đặc trưng (cường độ điểm ảnh RGB của ảnh).
 - Đặc trưng: "độ mềm" và "độ sáng" của lông và biểu diễn trục x là độ mềm và trục y là "độ sáng"
 - ➔ Khoảng cách giữa hai ảnh con mèo nhỏ hơn so với khoảng cách giữa một con mèo và một con chó.



- Dự đoán một "động vật chưa biết" là con gì?
- Xét trường hợp $k=1$, tức là xét 1 láng giềng gần nhất (1 con gần nhất)
- Với dữ liệu này, thấy rằng: Vì ảnh con vật đầu vào gần nhất với 01 điểm dữ liệu là dog → ảnh đầu vào được phân loại là dog



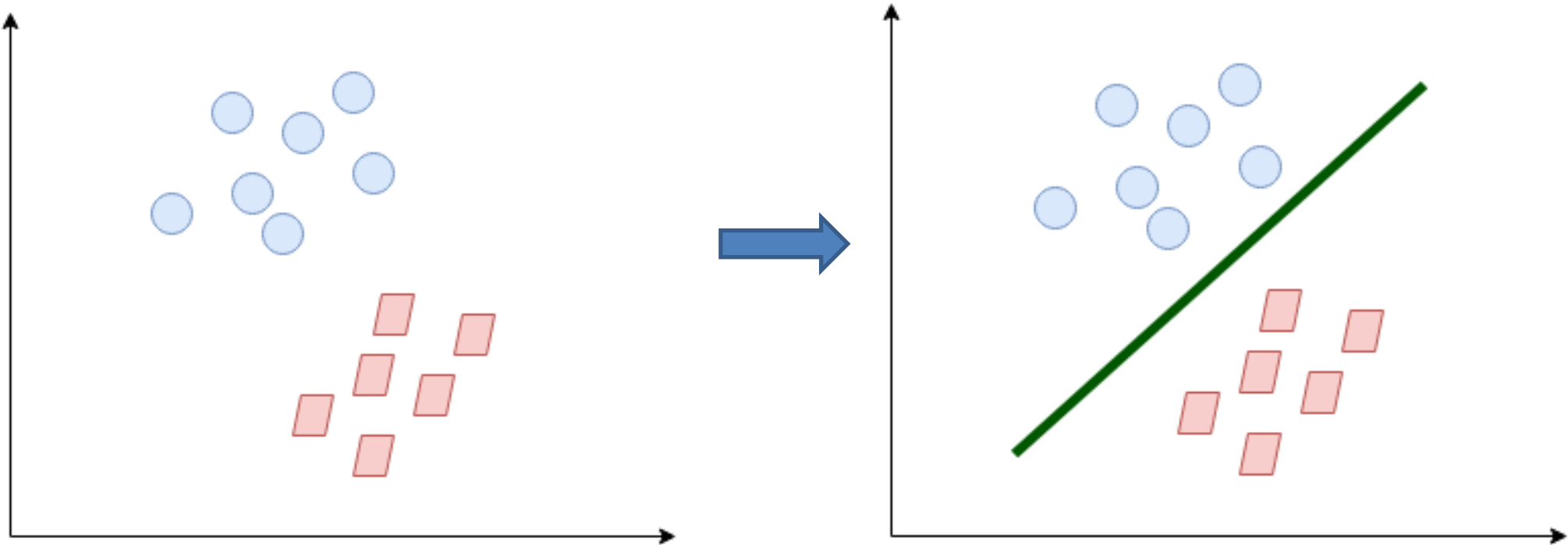
- Xét trường hợp $k = 3$ (3 con gần nhất)
- Thấy rằng: ảnh này gần nhất là 2 con cat và 1 con panda. Vì cat “nhiều hơn” → phân loại ảnh đầu vào là con cat
- Tiếp tục thực hiện với các giá trị khác nhau của k , ảnh con vật nào có số lượng lớn nhất trong k điểm huấn luyện gần nhất thì được gán nhãn cho ảnh đầu vào
- Lưu ý: Nếu k quá nhỏ → dễ bị nhiễu và các điểm dữ liệu ngoại lệ. Nếu k quá lớn, thì có nguy cơ làm mịn quá mức các kết quả phân loại và làm tăng độ lệch.
- Tham số k , d: gọi là siêu tham số Hyperparameter



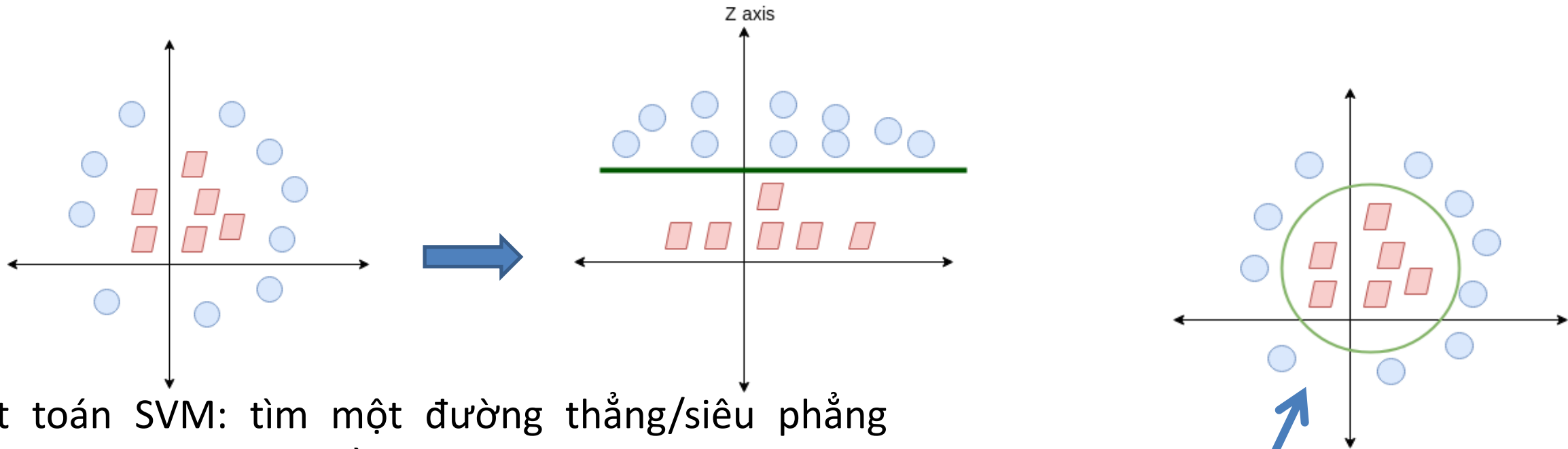
- **Tóm lại: Các bước thuật toán k-NN để phân loại ảnh** (Sử dụng thư viện sklearn)
 - Bước 1: Sưu tập dữ liệu
 - Bước 2: Chia tách dữ liệu
 - Bước 3: Tạo và huấn luyện bộ phân lớp: `model = KNeighborsClassifier(n_neighbors=n)`
`model.fit(cặp dữ liệu và nhãn để huấn luyện)`
 - Bước 4: Đánh giá: `model.predict(cặp dữ liệu và nhãn để đánh giá)`
 - Bước 5: Ứng dụng (Xác định nhãn cho dữ liệu mới)
- **Ưu và nhược thuật toán k-NN**
 - Đơn giản
 - Phù hợp với các không gian đặc trưng ít chiều
 - Không thực sự “học” bất cứ thứ gì
 - Nó chỉ đơn giản là dựa vào khoảng cách trong không gian n-chiều để phân loại

- **Thực hành các bước thực thi thuật toán k-NN để phân loại ảnh**
 - **Bước 1: Sưu tập dữ liệu**
 - Animal Dataset gồm: 3.000 ảnh RGB với 1.000 ảnh tương ứng cho mỗi lớp dog, cat và panda.
 - Tiền xử lý từng ảnh bằng cách thay đổi kích thước của nó thành 32 x 32 pixel → với 3 kênh RGB thì kích thước ảnh trong Dataset là $32 \times 32 \times 3 = 3072$
 - **Bước 2: Chia tách dữ liệu:** Chia dữ liệu trong Dataset hai phần: phần để train và phần để test
 - **Bước 3: Huấn luyện bộ phân lớp:** Bộ phân loại k-NN sẽ được đào tạo về cường độ pixel của các ảnh trong tập train và lưu mô hình thành file
 - **Bước 4: Đánh giá:** Sử dụng dữ liệu test để đánh giá hiệu suất của bộ phân lớp k-NN
 - **Bước 5: Ứng dụng:** Sử dụng model đã huấn luyện để viết ứng dụng phân loại ảnh

- Giả sử có một biểu đồ gồm 02 lớp → Sử dụng đường thẳng để tách
- → Phương pháp tuyến tính



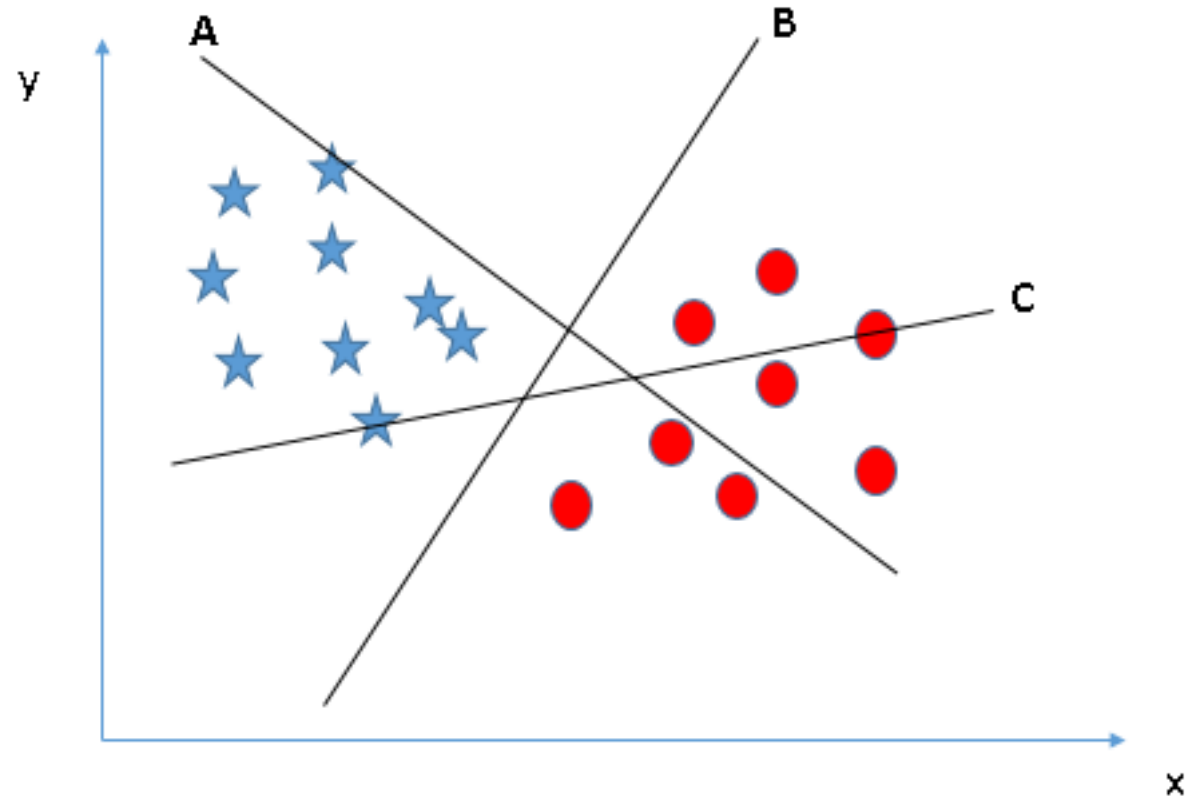
- Bây giờ dữ liệu phân bố như sau → Thì làm sao tách?
- Chuyển sang một không gian biểu diễn mới → mở rộng thêm chiều z → tách dễ dàng



Thuật toán SVM: tìm một đường thẳng/siêu phẳng (trong không gian đa chiều) ngăn cách hai lớp này → nó phân loại điểm dữ liệu mới tùy thuộc vào việc điểm dữ liệu mới này nằm ở phía dương hay âm của đường thẳng/siêu phẳng để dự đoán

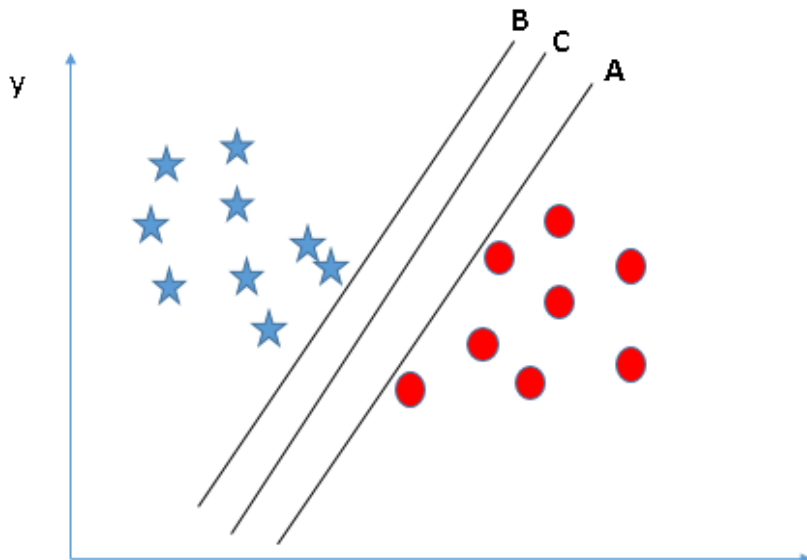
Nếu chuyển về không gian ban đầu

- **Kịch bản 1:** Xác định siêu phẳng đúng
 - Có 3 siêu phẳng (A, B và C). Hãy xác định siêu phẳng phù hợp để phân loại các ngôi sao và vòng tròn
 - Chọn siêu phẳng nào phân tách hai lớp tốt hơn ?
 - ➔ Trong kịch bản này, siêu phẳng “B” đã thực hiện xuất sắc nhiệm vụ này



- Kịch bản 2:** Xác định siêu phẳng đúng

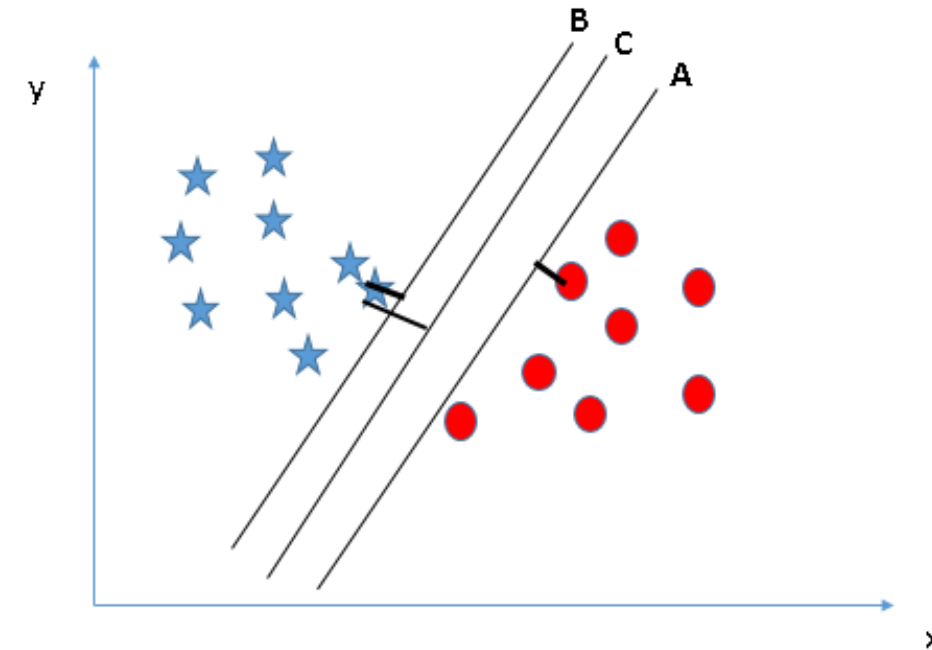
- Có ba siêu phẳng (A, B và C) và tất cả đều phân lớp tốt → Làm thế nào để xác định đúng siêu phẳng?



→ Tìm khoảng cách lớn nhất giữa điểm dữ liệu gần nhất (của các lớp) và siêu phẳng sẽ giúp quyết định siêu phẳng phù hợp.

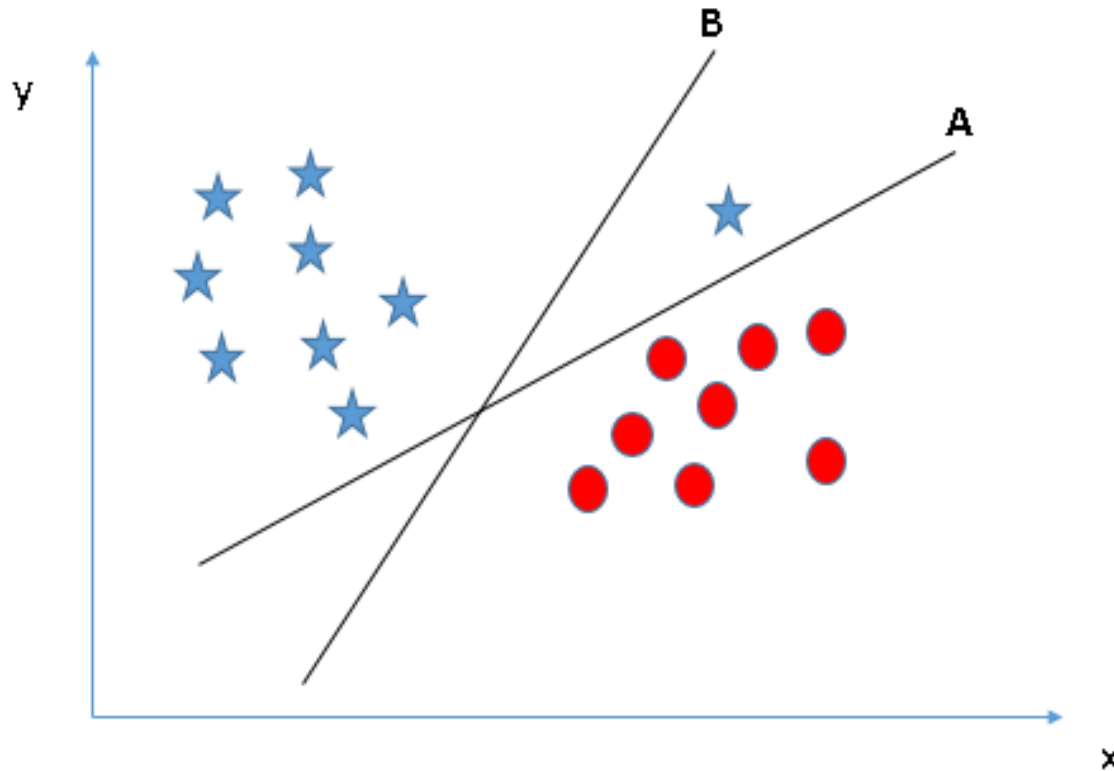
→ Khoảng cách này được gọi là **Margin**

→ Các điểm dữ liệu gần nhất này gọi là Vec tơ hỗ trợ (Support Vector)



Margin của siêu phẳng C lớn hơn cả A và B → chọn siêu phẳng đúng là C

- Kịch bản 3: Xác định siêu phẳng đúng



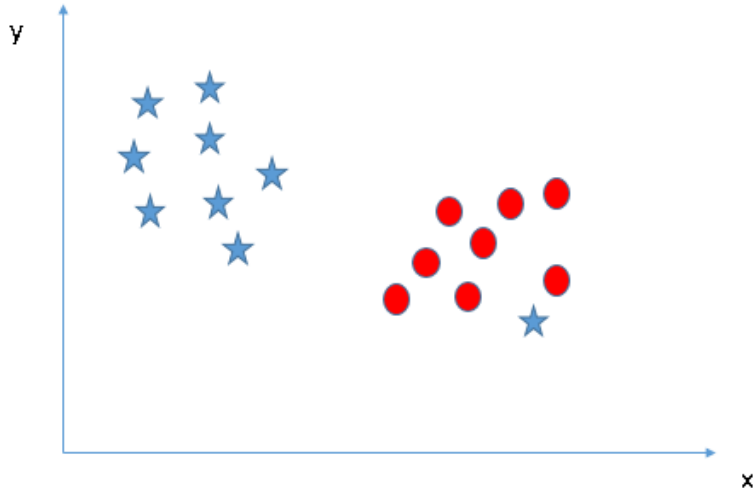
→ Chọn siêu phẳng B vì nó có Margin cao hơn siêu phẳng A ?

SVM ưu tiên chọn siêu phẳng phân loại chính xác trước khi sử dụng phương pháp tối đa hóa Margin

→ Siêu phẳng B có lỗi phân loại

→ A đã phân loại đúng → Chọn siêu phẳng bên đúng là A.

- Kịch bản 4: Xác định siêu phẳng đúng**



Sử dụng đường thẳng để phân loại

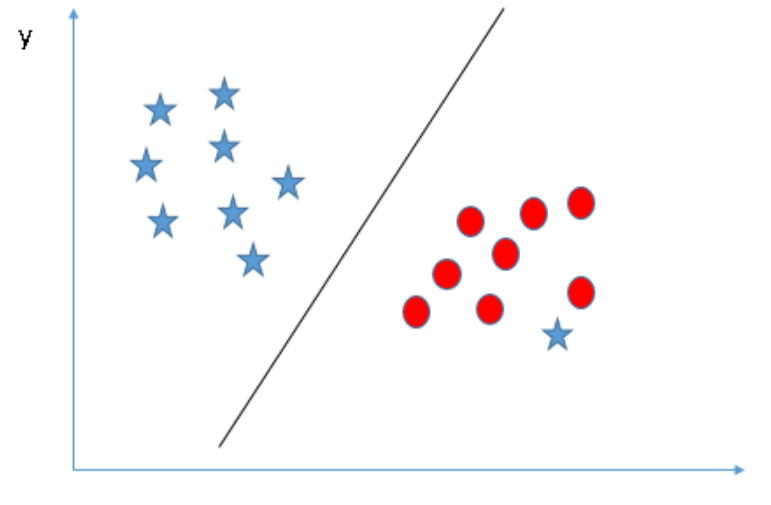
→ Gọi là siêu phẳng tuyến tính

→ Có những trường hợp không thể dùng siêu phẳng tuyến tính để phân loại

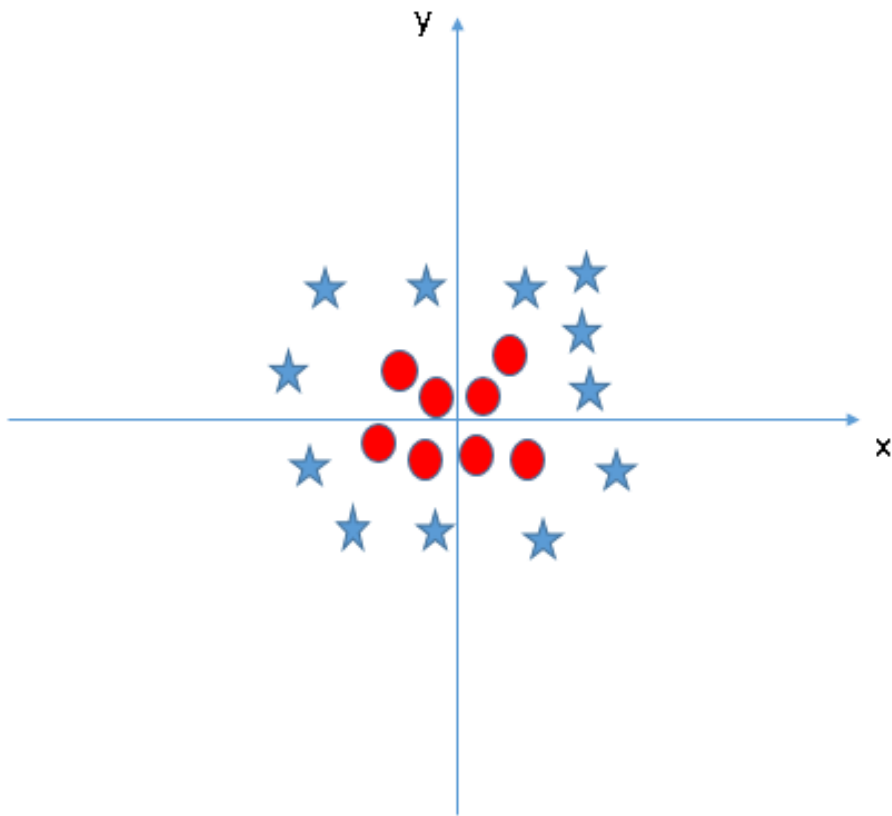
→ Chọn siêu phẳng nào đúng để phân loại?

Ngôi sao ở vùng đường tròn gọi là ngoại lệ

→ SVM sử dụng phương pháp tối đa hóa Margin để phân loại → dùng đường thẳng



- Kịch bản 5: Xác định siêu phẳng đúng



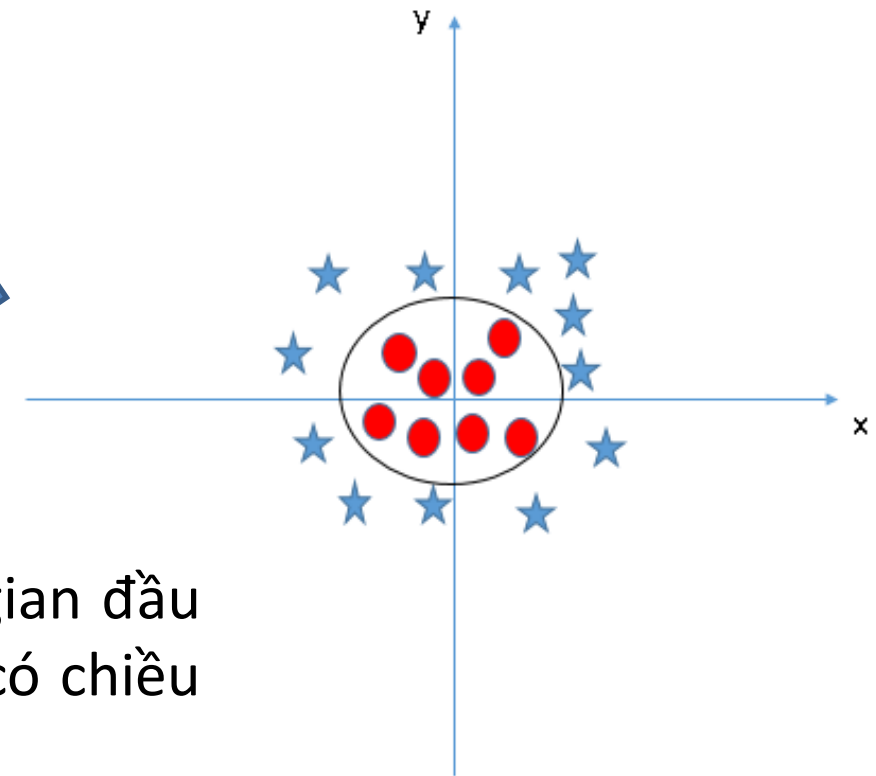
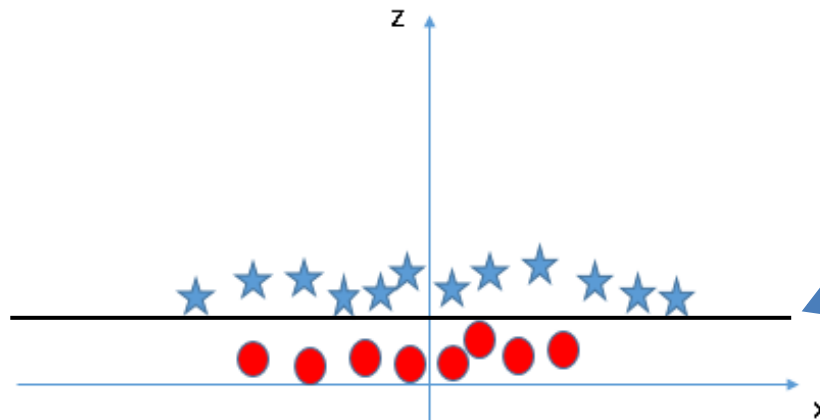
- ➔ Chọn siêu phẳng nào đúng để phân loại?
- ➔ Sử dụng siêu phẳng phi tuyến
- ➔ Chuyển không gian biểu diễn dữ liệu
- ➔ Ví dụ: $z = x^2 + y^2$

- Kịch bản 5: Xác định siêu phẳng đúng

→ Sử dụng siêu phẳng phi tuyến

→ Chuyển không gian biểu diễn dữ liệu

→ Ví dụ: $z = x^2 + y^2$



SVM sử dụng kỹ thuật gọi là **kernel trick** lấy không gian đầu vào có chiều thấp và biến đổi nó thành không gian có chiều cao hơn

- Các hàm kernel thường dùng

- Linear: Tích vô hướng của 2 vector $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
 - Trong thư viện sklearn thì kernel này được chọn bằng cách đặt `kernel = 'linear'`
- Polynomial: $k(\mathbf{x}, \mathbf{z}) = (r + \gamma \mathbf{x}^T \mathbf{z})^d$
 - Trong thư viện sklearn thì dùng `kernel = 'poly'`
- Radial Basic Function (RBF): $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2)$, $\gamma > 0$
 - Trong thư viện sklearn thì dùng `kernel = 'rbf'`
- Sigmoid: $k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$
 - Trong thư viện sklearn thì dùng `kernel = 'sigmoid'`

- Khoảng cách

- Trong không gian 2 chiều, khoảng cách từ một điểm có tọa độ (x_0, y_0) tới **đường thẳng** có phương trình $w_1x + w_2y + b = 0$ được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

- Trong không gian 3 chiều, khoảng cách từ một điểm có tọa độ (x_0, y_0, z_0) tới một **mặt phẳng** có phương trình $w_1x + w_2y + w_3z + b = 0$ được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

- Trong không gian d chiều: Khoảng cách từ một điểm (vector) có tọa độ x_0 tới **siêu mặt phẳng (hyperplane)** có phương trình $w^T x_0 + b = 0$ được xác định bởi:

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

Với $\|w\|_2 = \sqrt{\sum_{i=1}^d w_i^2}$ với d là số chiều của không gian.

• Thuật toán SVM

- Giả sử các cặp dữ liệu **training** là $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$
- Vector $x_i \in \mathbb{R}^d$ thể hiện **đầu vào** của một điểm dữ liệu (ở đây là ảnh đầu vào)
- y_i là **nhãn** của điểm dữ liệu đó (nhãn của ảnh)
- d là số chiều của dữ liệu (đối với ảnh RGB có 3 chiều/kênh)
- N là số điểm dữ liệu (số lượng ảnh)
- Với cặp (x_n, y_n) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là: $\frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|_2}$

Khoảng cách từ điểm dữ liệu gần nhất $= \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|_2}$

- **Xây dựng bài toán tối ưu cho SVM (tiếp)**

→ Mục tiêu tìm Margin, tức là tìm giá trị của w và b để “Khoảng cách từ điểm dữ liệu gần nhất” là lớn nhất

$$(\mathbf{w}, b) = \arg \max_{\mathbf{w}, b} \left\{ \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|_2} \right\} = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n(\mathbf{w}^T \mathbf{x}_n + b) \right\}$$

- **Xác định nhãn cho một điểm dữ liệu mới:**

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

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

→ Ví dụ: Đối với bài toán phân lớp nhị phân (2 lớp) thì $\text{sgn}()$ là hàm đổi dấu: bằng 1 khi đối số là không âm và -1 ngược lại

- **Tóm lại: Các bước thuật toán SVM để phân loại ảnh**

- Bước 1: Sưu tập dữ liệu
- Bước 2: Chia tách dữ liệu
- Bước 3: Tạo và huấn luyện bộ phân lớp: `model = svm.SVC(kernel)`
`model.fit(cặp dữ liệu và nhãn để huấn luyện)`
- Bước 4: Đánh giá: `model.predict(cặp dữ liệu và nhãn để đánh giá)`
- Bước 5: Ứng dụng (Xác định nhãn cho dữ liệu mới)

- **Thực hành các bước thực thi thuật toán SVM để phân loại ảnh**
 - **Bước 1: Sưu tập dữ liệu**
 - Animal Dataset gồm: 3.000 ảnh RGB với 1.000 ảnh tương ứng cho mỗi lớp dog, cat và panda.
 - Tiền xử lý từng ảnh bằng cách thay đổi kích thước của nó thành 32 x 32 pixel → với 3 kênh RGB thì kích thước ảnh trong Dataset là $32 \times 32 \times 3 = 3072$
 - **Bước 2: Chia tách dữ liệu:** Chia dữ liệu trong Dataset hai phần: phần để train và phần để test
 - **Bước 3: Tạo và Huấn luyện bộ phân lớp và lưu mô hình thành file**
 - **Bước 4: Đánh giá:** Sử dụng dữ liệu test để đánh giá hiệu suất của bộ phân lớp SVM
 - **Bước 5: Ứng dụng:** Sử dụng model đã huấn luyện để viết ứng dụng phân loại ảnh

- **Bài tập: Huấn luyện bộ phân loại kNN và SVM và viết ứng dụng để phân loại cho các yêu cầu sau:**
 - Bài 1. Phân loại 10 loài hoa
 - Bài 2. Phân loại 10 loài vật
 - Bài 3. Phân loại 10 ký tự số (0 đến 9)
 - Bài 4. Phân loại 10 ký hiệu bảng chỉ dẫn giao thông
 - Bài 5. Phân loại 03 tín hiệu đèn (Vàng, Đỏ, Xanh)
 - Bài 6. Phân loại 10 loại quả
 - Bài 7. Phân loại 10 loại thức ăn
 - Bài 8. Phân loại trang phục của 10 đôi bóng đá
 - Bài 9. Phân loại 03 cầu thủ (Công phượng, Quang Hải, Tuấn Anh)
 - Bài 10. Phân loại 10 bạn sinh viên trong lớp