# BÁO CÁO LÝ THUYẾT – SUPPORT VECTOR MACHINE (SVM)

## 1️⃣ Giải thuật Support Vector Machine hoạt động như thế nào?

SVM (Support Vector Machine) là thuật toán phân loại có giám sát, nhằm tìm ra siêu phẳng (hyperplane) tối ưu để phân tách các lớp dữ liệu. SVM chọn đường biên có khoảng cách (margin) lớn nhất đến các điểm gần nhất của mỗi lớp.

• Hyperplane: Ranh giới phân tách giữa hai lớp dữ liệu.

• Margin: Khoảng cách từ siêu phẳng đến các điểm dữ liệu gần nhất. SVM cố gắng tối đa hóa khoảng cách này để tăng khả năng tổng quát hóa.

## 2️⃣ Các vector hỗ trợ (Support Vectors) và vai trò của chúng

Support Vectors là các điểm dữ liệu nằm gần ranh giới phân tách nhất. Chúng quyết định vị trí và hướng của siêu phẳng tối ưu. Những điểm này có ảnh hưởng trực tiếp đến mô hình, trong khi các điểm xa hơn thì không.

## 3️⃣ Sự khác biệt giữa SVM với lề cứng (Hard Margin) và lề mềm (Soft Margin)

|  |  |  |
| --- | --- | --- |
| Đặc điểm | Hard Margin | Soft Margin |
| Định nghĩa | Không cho phép sai số, tất cả điểm phải được phân tách chính xác. | Cho phép một số điểm sai để mô hình tổng quát hơn. |
| Dữ liệu yêu cầu | Phân tách tuyến tính hoàn toàn. | Không phân tách tuyến tính hoàn toàn. |
| Rủi ro | Dễ overfit khi có nhiễu. | Ổn định hơn, ít bị overfit. |

👉 Nên sử dụng Soft Margin khi dữ liệu có nhiễu hoặc không thể phân tách tuyến tính hoàn toàn.

## 4️⃣ Hàm nhân (Kernel) trong SVM

Khi dữ liệu không thể phân tách tuyến tính, SVM sử dụng hàm nhân (Kernel Function) để ánh xạ dữ liệu lên không gian có chiều cao hơn, tại đó có thể tìm được siêu phẳng phân tách tuyến tính.

|  |  |  |
| --- | --- | --- |
| Kernel | Công thức / Tên | Khi nên dùng |
| Linear | K(x, y) = x · y | Dữ liệu có thể phân tách tuyến tính. |
| Polynomial | K(x, y) = (x · y + c)^d | Dữ liệu có quan hệ bậc cao giữa các đặc trưng. |
| RBF | K(x, y) = exp(-γ ||x−y||²) | Dữ liệu phi tuyến, phổ biến nhất. |
| Sigmoid | K(x, y) = tanh(αx·y + c) | Ít dùng, tương tự mạng nơ-ron. |

## 5️⃣ Tham số C trong SVM

Tham số C điều chỉnh mức độ phạt cho các điểm bị sai. Nó kiểm soát sự cân bằng giữa việc tối đa hóa lề và giảm sai số huấn luyện.

|  |  |  |
| --- | --- | --- |
| Giá trị C | Ý nghĩa | Ảnh hưởng |
| C lớn | Ít cho phép sai số. | Lề nhỏ, dễ overfit. |
| C nhỏ | Cho phép sai số nhiều hơn. | Lề lớn, tổng quát hơn. |

## 6️⃣ Code mẫu xây dựng mô hình SVM (Python, Scikit-learn)

from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score  
  
# 1. Tải dữ liệu  
iris = datasets.load\_iris()  
X, y = iris.data, iris.target  
  
# 2. Chia dữ liệu  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 3. Chuẩn hóa dữ liệu  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
# 4. Tạo và huấn luyện mô hình SVM  
model = SVC(kernel='rbf', C=1.0, gamma='scale', random\_state=42)  
model.fit(X\_train\_scaled, y\_train)  
  
# 5. Dự đoán và đánh giá  
y\_pred = model.predict(X\_test\_scaled)  
print("Độ chính xác:", accuracy\_score(y\_test, y\_pred))

## 7️⃣ Chuẩn hóa dữ liệu trong SVM

Hàm dùng để chuẩn hóa: StandardScaler() trong sklearn.preprocessing.

from sklearn.preprocessing import StandardScaler  
  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)

👉 Việc chuẩn hóa rất quan trọng vì SVM dựa trên khoảng cách Euclidean giữa các điểm. Nếu đặc trưng có thang đo khác nhau, các giá trị lớn sẽ chi phối mô hình, khiến SVM hoạt động kém hiệu quả.