## Thuật toán phân loại Support Vector Machine

Trong một hệ thống nhận dạng khuôn mặt, việc định danh và phân loại khuôn mặt là một bước vô cũng quan trọng. Ở bước này, bằng việc sử dụng các thuật toán dự đoán đầu ra với Machine Learning hay Deep Learning, hệ thống sẽ thực hiện việc phân loại ảnh khuôn mặt, hay Vector dữ liệu có được từ PCA, thuộc vào lớp nào đã được định nghĩa từ trước. Có khá nhiều phương pháp phân loại phổ biến như Convolutional Neural Network, hay Support Vector Machine. Sau quá trình thực hiện thử nghiệm các thuật toán phân loại khác nhau cũng như xem xét ưu nhược điểm của từng thuật toán, tôi quyết định sử dụng thuật toán phân loại SVM cho hệ thống này.

### SVM và các phương pháp liên quan

SVM là một thuật toán thuộc nhóm Supervised Learning – học có giám sát, dùng để phân chia dữ liệu thành các nhóm riêng biệt.

Giả sử có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai class này *linearly separable* (phân biệt tuyến tính), sẽ tồn tại các siêu mặt phẳng phân chia hai class đó.

Chart, scatter chart

Description automatically generated

*Hình 2.4: Các siêu mặt phẳng phân chia hai class linearly separable [5]*

Siêu mặt phẳng là mặt phẳng quyết định việc phân chia các điểm dữ liệu vào các class khác nhau. *Support Vectors* là điểm dữ liệu của mỗi class mà gần với siêu mặt phẳng nhất. Và *margin* là khoảng cách từ *Support Vectors* đến siêu mặt phẳng.

Hình 2.5, hình đầu cho thấy siêu mặt phẳng không tối ưu vì khoảng cách của nó đến hai Support Vectors của hai class là không bằng nhau, tức margin không bằng nhau. Với hình thứ hai của hình 2.5, siêu mặt phẳng với nét đứt màu xanh có margin không rộng bằng siêu mặt phẳng với nét liền màu đen. Vậy siêu mặt phẳng với nét đứt màu đen là siêu mặt phẳng tối ưu cần tìm.

Diagram, scatter chart

Description automatically generatedDiagram

Description automatically generated

*Hình 2.5: Xác định siêu mặt phẳng tối ưu bằng Support Vectors và Margin [5]*

Bài toán tối ưu trong SVM chính là việc đi tìm siêu mặt phẳng sao cho margin đến hai class là bằng nhau và lớn nhất.

Ví dụ ở trên là trường hợp đơn giản nhất đối với SVM, phân loại dữ liệu *linearly separable*, đây còn gọi là phương pháp SVM thuần, hay **Hard Margin SVM**.

Với hình 2.6a, dữ liệu vẫn *linearly separable*, nhưng có một điểm nhiễu của lớp tròn đỏ ở quá gần so với lớp vuông xanh. Trong trường hợp này, nếu ta sử dụng SVM thuần thì sẽ tạo ra một margin rất nhỏ. Ngoài ra, đường phân lớp nằm quá gần lớp vuông xanh và xa lớp tròn đỏ. Trong khi đó, nếu ta hy sinh điểm nhiễu này thì ta được một margin tốt hơn rất nhiều được mô tả bởi các đường nét đứt. SVM thuần vì vậy còn được coi là nhạy cảm với nhiễu.

Với hình 2.6b, dữ liệu không *linearly separable* nhưng gần *linearly separable*, trường hợp này nếu ta sử dụng SVM thuần thì rõ ràng bài toán tối ưu là vô nghiệm. Tuy nhiên, nếu ta lại chịu hy sinh một chút những điểm ở gần biên giữa hai class, ta vẫn có thể tạo được một đường phân chia khá tốt như đường nét đứt đậm. Các đường support nét đứt mảnh qua các Support Vector vẫn giúp tạo được một margin lớn cho bộ phân lớp này. Với mỗi điểm nằm lần sang phía bên kia của các đường support tương ứng, ta gọi điểm đó rơi vào vùng không an toàn. Vùng an toàn của hai class là khác nhau, giao nhau ở phần nằm giữa hai đường support.

Hai trường hợp dựa vào hình 2.6 cho ta hai siêu mặt phẳng với các đường support được gọi là soft margin. Việc hy sinh một vài các điểm dữ liệu, nhằm thay đổi các Support Vector thông thường thành các điểm dữ liệu ở xa hơn, để tạo ra một siêu mặt phẳng tối ưu hơn, đây là phương pháp **Soft margin SVM**.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

*Hình 2.6: Trường hợp Soft margin SVM [5]*

Đối với dữ liệu hoàn toàn không *linearly separable*, có một phương pháp gọi là **Kernel SVM** để giải quyết bài toán này.

Chart, scatter chart

Description automatically generated

*Hình 2.7: Dữ liệu không linearly separable [5]*

Ý tưởng cơ bản của Kernel SVM là việc đi tìm một phép biến đổi sao cho dữ liệu ban đầu là không phân biệt tuyến tính được biến đổi sang không gian mới. Ở không gian mới này, dữ liệu trở nên phân biệt tuyến tính.

Hình 2.7 cho ta ví dụ về dữ liệu không *linearly separable*trong không gian hai chiều. Đối với dữ liệu như vậy không thể tìm được siêu mặt phẳng dựa vào cả Hard margin SVM và Soft Margin SVM.

Hình 2.8 cho ta thấy phương pháp để giải quyết bài toán với dữ liệu không *linearly separable.* Cách giải quyết của Kernel SVM là đi tìm một hàm biến đổi dữ liệu từ không gian đặc trưng ban đầu thành dữ liệu trong một không gian mới bằng hàm số . Trong ví dụ này, hàm là việc thêm một chiều dữ liệu mới – đặc trưng mới là một hàm số của các đặc trưng đã biết. Hàm số này thỏa mãn mục đích của chúng ta: trong không gian mới này, dữ liệu của hai class là phân biệt tuyến tính hay gần phân biệt tuyến tính. Khi đó, ta có thể sử dụng Hard/Soft margin SVM để giải quyết bài toán.

Chart

Description automatically generatedChart, radar chart, scatter chart

Description automatically generated

*Hình 2.8: Ý tưởng Kernel SVM [5]*

Các hàm kernel được sử dụng trong Kernel SVM là: linear, polynomial (poly), radial basic function (rbf), sigmoid.

Trong hệ thống này của tôi, dựa trên thực nghiệm cũng như tính chất của dữ liệu đầu vào, tôi sử dụng Kernel SVM để giải quyết bài toán phân loại.

### SVM và bài toán phân loại nhiều lớp

Các phương pháp SVM được đề cập ở trên – Soft margin SVM, Hard margin SVM, Kernel SVM đều được xây dựng nhằm giải quyết bài toán Binary Classification, phân biệt chỉ 2 lớp. Để mở rộng các mô hình Binary classifier áp dụng được cho các bài toán multi-class classification, ta sử dụng các phương pháp như one-vs-one hay one-vs-rest.

**One-vs-one** là phương pháp xây dựng rất nhiều bộ binary classifier cho từng cặp class. Bộ thứ nhất phân biệt class 1 với class 2, bộ thứ 2 phân biệt class 1 với class 3, … Khi có một dữ liệu mới vào, đưa nó vào toàn bộ các binary classifier trên. Kết quả cuối cùng có thể xác định bằng cách xem class nào mà điểm dữ liệu đó được phân vào nhiều nhất. Như vậy, nếu có class, tổng số binary classifier cần phải dùng là . Đây là một con số lớn, không lợi về mặt tính toán. Hơn nữa, nếu một điểm dữ liệu là class 1, nhưng lại được đưa vào bộ phân lớp cho class 2 và class 3, thì cả hai khả năng đều không hợp lý.

Phương pháp được sử dụng phổ biến hơn là **One-vs-rest**. Cụ thể, nếu có class, ta sẽ xây dựng bộ phân lớp, mỗi bộ phân lớp tương ứng với một class. Bộ phân lớp thứ nhất phân biệt class 1 với not-class 1, tức là xem một điểm dữ liệu có phải là class 1 hay không. Tương tự như thế, bộ phân lớp thứ 2 phân biệt class 2 với not-class 2, … Kết quả cuối cùng có thể xác định bằng cách tìm class mà xác xuất rơi vào cao nhất, hay với SVM là khoảng cách điểm đó đến siêu mặt phẳng là lớn nhất.

Trong hệ thống này, tôi sử dụng phương pháp One-vs-rest để giải bài toán Multi Classification cho Kernel SVM.

### Thông số mô hình SVM của hệ thống

Hai thông số quan trọng trong một mô hình rbf Kernel SVM là và **γ**.

là thông số giúp cân đối độ lớn của margin và sự hy sinh của các điểm nằm trong vùng không an toàn. Khi hoặc rất lớn, Soft margin SVM trở thành Hard margin SVM. Tham số nhỏ làm margin lớn hơn, từ đó làm đơn giản hàm quyết định. Nói cách khác, hoạt động như một tham số regularization trong SVM, ảnh hưởng đến việc overfitting trong mô hình.

Tham số xác định mức độ ảnh hưởng của mỗi mẫu trong bộ train, giá trị thấp nghĩa là “xa” và giá trị cao nghĩa là “gần”. Tham số có thể xem là nghịch đảo bán kính ảnh hưởng của các điểm dữ liệu mà mô hình chọn là Support Vectors.