1. Giới thiệu

Có một sự tương ứng thú vị giữa hai nhóm thuật toán phân lớp phổ biến nhất: Neural Network và Support Vector Machine (SVM). Chúng đều bắt đầu từ bài toán phân lớp với 2 *linearly separable classes*, tiếp theo đến 2 *almost linear separable classes (2 lớp có thể phân tách gần như tuyến tính)*, đến bài toán có nhiều classes rồi các bài toán với biên không tuyến tính. Sự tương ứng được cho trong bảng dưới đây:

| **Neural Networks** | **Support Vector Machine** | **Tính chất chung** |
| --- | --- | --- |
| [PLA](https://machinelearningcoban.com/2017/01/21/perceptron/) | [Hard Margin SVM](https://machinelearningcoban.com/2017/04/09/smv/) | Hai classes là *linearly separable* |
| [Logistic Regression](https://machinelearningcoban.com/2017/01/27/logisticregression/) | [Soft Margin SVM](https://machinelearningcoban.com/2017/04/13/softmarginsmv/) | Hai classes là *gần linearly separable* |
| [Softmax Regression](https://machinelearningcoban.com/2017/02/17/softmax/) | Multi-class SVM | Bài toán phân loại nhiều classes (biên là tuyến tính) |
| [Multi-layer Perceptron](https://machinelearningcoban.com/2017/02/24/mlp/) | Kernel SVM | Bài toán với dữ liệu không *linearly separable* |

Kernel SVM: áp dụng SVM lên bài toán mà dữ liệu giữa 2 classes là hoàn toàn không linear separable.

Ý tưởng cơ bản: tìm 1 phép biến đổi sao cho dữ liệu ban đầu không phân biệt tuyến tính được biến sang không gian mới. Ở không gian mới này , dữ liệu trở nên phân biệt tuyến tính.

Xét ví dụ dưới đây với việc biến dữ liệu không phân biệt tuyến tính trong không gian hai chiều thành phân biệt tuyến tính trong không gian ba chiều bằng cách giới thiệu thêm một chiều mới:

Chart

Description automatically generated with low confidence

Tóm gọn thì kernel SVM là đi tìm 1 hàm số biến đổi dữ liệu x từ không gian ban đầu thành dữ liệu trong 1 không gian mới bằng hàm số phi(x). Trong vd này thì hàm phi() đơn giản là hàm giới thiệu 1 chiều dữ liệu mới (1 feature mới) là 1 hàm số của các feature đã biết

* Hàm số này cần phải thỏa 1 số điều kiện, mục đích: trong không gian mới thì dữ liệu giữa 2 classes là phân biệt tuyến tính hoặc là gần tuyến tính. Để khi đó ta dùng được các bộ phân lớp tuyến tính thông thường như PLA, logistic regression, hard/soft margin svm

Nếu phải so sánh, ta có thể thấy rằng hàm biến đổi Φ() tương tự như [*activation functions*](https://machinelearningcoban.com/2017/02/24/mlp/#-activation-functions) trong Neural Networks. Tuy nhiên, có một điểm khác biệt ở đây là: trong khi nhiệm vụ của activation function là phá vỡ tính tuyến tính của *mô hình*, hàm biến đổi Φ() đi biến *dữ liệu* không phân biệt tuyến tính thành phân biệt tuyến tính. Như vậy là để đạt được mục đích chung, ta có hai cách nhìn khác nhau về cách giải quyết.

Các hàm Φ() thường tạo ra dữ liệu mới có số chiều cao hơn số chiều của dữ liệu ban đầu, thậm chí là vô hạn chiều. Nếu tính toán các hàm này trực tiếp, chắc chắn chúng ta sẽ gặp các vấn đề về bộ nhớ và hiệu năng tính toán. Có một cách tiếp cận là sử dụng các *kernel functions* mô tả quan hệ giữa hai điểm dữ liệu bất kỳ trong không gian mới, thay vì đi tính toán trực tiếp từng điểm dữ liệu trong không gian mới. Kỹ thuật này được xây dựng dựa trên quan sát về [bài toán đối ngẫu của SVM](https://machinelearningcoban.com/2017/04/09/smv/#-bai-toan-doi-ngau-cho-svm).

3.2. Một số hàm kernel thông dụng

3.2.1. Linear

Đây là trường hợp đơn giản với kernel chính tích vô hướng của hai vector:

Graphical user interface, text

Description automatically generated

Với d là một số dương để chỉ bậc của đa thức. d có thể không là số tự nhiên vì mục đích chính của ta không phải là bậc của đa thức mà là cách tính kernel. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.

Phần kiểm tra liệu hàm này có thỏa mãn điều kiện (7)(7) hay không xin được bỏ qua.

Khi sử dụng thư viện sklearn, kerrnel này được chọn bằng cách đặt kernel = 'poly'. Thông tin cụ thể về cách sử dụng có thể xem [tại đây](http://scikit-learn.org/stable/modules/svm.html#svm-kernels).

3.2.3. Radial Basic Function

Radial Basic Function (RBF) kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế, và là lựa chọn mặc định trong sklearn. Nó được định nghĩa bởi:

Table

Description automatically generated

5. Tóm tắt

* Nếu dữ liệu của hai lớp là *không phân biệt tuyến tính*, chúng ta có thể tìm cách biến đổi dữ liệu sang một không gian mới sao cho trong không gian mới ấy, dữ liệu của hai lớp là *phân biệt tuyến tính* hoặc *gần phân biệt tuyến tính*.
* Việc tính toán trực tiếp hàm Φ() đôi khi phức tạp và tốn nhiều bộ nhớ. Thay vào đó, ta có thể sử dụng **kernel trick**. Trong cách tiếp cận này, ta chỉ cần tính tích vô hướng của hai vector bất kỳ trong không gian mới: k(x,z)=Φ(x)TΦ(z)
* Thông thường, các hàm k() thỏa mãn điều kiện Merrcer, và được gọi là *kernel*. Cách giải bài toán SVM với kernel hoàn toàn giống với cách giải bài toán Soft Margin SVM.
* Có 4 loại kernel thông dụng: linear, poly, rbf, sigmoid. Trong đó, rbf được sử dụng nhiều nhất và là lựa chọn mặc định trong các thư viện SVM.
* Với dữ liệu *gần phân biệt tuyến tính*, linear và poly kernels cho kết quả tốt hơn.

Ta có các nhận xét đối với mỗi kernel như sau:

* sigmoid: nghiệm tìm được không thật tốt vì có 3 trong 4 điểm nằm chính xác trên đường phân chia. Nói cách khác, nghiệm này rất nhạy cảm với nhiễu.
* poly: Nghiệm này có tốt hơn nghiệm của sigmoid nhưng kết quả có phần giống với [overfitting](https://machinelearningcoban.com/2017/03/04/overfitting/).
* rbf: Dữ liệu được tạo ra một cách đối xứng, đường phân lớp tìm được cũng tạo ra các vùng đối xứng với mỗi class. Nghiệm này được cho là hợp lý hơn. Trên thực tế, các rbf kernel được sử dụng nhiều nhất và cũng là lựa chọn mặc định trong hàm sklearn.svm.SVC.

***Dữ liệu gần linearly seperable***

Chart

Description automatically generated