**PERCEPTRON LEARNING ALGORITHM (PLA)**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**TOÁN KHOA HỌC MÁY TÍNH**

**CS115.O12**

**Giảng viên: Lương Ngọc Hoàng, Dương Việt Hằng**

**Sinh viên: Võ Đại Lượng**

**MSSV: 22520834**

**MỤC LỤC:**

[1. Giới Thiệu Perceptron Learning Algorithm (PLA). 2](#_Toc147346510)

[1.1. Perceptron Learning Algorithm (PLA) là gì? 2](#_Toc147346511)

[1.2. Ý tưởng cơ bản của PLA? 2](#_Toc147346512)

[1.3 Bải toán Perceptron. 2](#_Toc147346513)

[2. Thuật toán Perceptron. 3](#_Toc147346514)

[2.1. Xây dựng hàm mất mát. 3](#_Toc147346515)

[2.2. Chứng minh hội tụ. 4](#_Toc147346516)

[2.3. Tóm tắt PLA và một vài lưu ý. 5](#_Toc147346517)

# 1. Giới Thiệu Perceptron Learning Algorithm (PLA).

## 1.1. Perceptron Learning Algorithm (PLA) là gì?

Perceptron Learning Algorithm (PLA) hay còn được viết gọn là Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất cho bài toán Binary Classification (tức bài toán chỉ có hai class) và trong một trường hợp rất cụ thể.

Perceptron là nền tảng cho một mảng lớp rất quan trọng trong Machine learning: Neural Network và sâu hơn nữa là Deep Learning.

## 1.2. Ý tưởng cơ bản của PLA?

Ý tưởng cơ bản của PLA: tìm lãnh thổ của mỗi class, sao cho với mỗi một điểm mới cần xác định nó nằm vào lãnh thổ của class nào rồi quyết định nó thuộc class đó.

Để tìm lãnh thổ của mỗi class, ta cần tìm boundary (biên giới) giữa hai lãnh thổ. Boundary có thể là một đường thẳng, mặt phẳng hoặc siêu mặt phẳng trong các chiều không gian khác nhau. Những boundary này có thể biểu diễn dưới dạng một hàm số đơn giản dưới dạng tuyến tính (linear).

## 1.3 Bải toán Perceptron.

Từ các ý trên, có thể phát biểu một bài toán Perceptron như sau: Cho hai class được gán nhãn, tìm một đường phẳng sao cho các điểm thuộc class 1 về 1 phía, vè các điểm thuộc class 2 nằm về phía còn lại với giả định rằng tồn tại một đường phẳng như vậy.

\* Nếu tồn tại một đường phẳng phân chia hai class thì ta gọi hai class đó là linearly separable, các thuật toán classication tạo ra các boundary là đường phẳng gọi chung là Classfier.

# 2. Thuật toán Perceptron.

Tương tự các thuật toán lặp (K-means Clustering, Gradient Descent, …), ý tưởng cơ bản của PLA là xuất phát từ một nghiệm dự đoán nào đó, qua mỗi vòng lặp, nghiệm sẽ được cập nhật tới một vị trí tốt hơn (việc cập nhật này đựa trên việc giảm giá trị của một hàm mất mát nào đó).

Giả sử mỗi điểm dữ liệu có số chiều là d=2, có đường thẳng:

***w1x1 + w2 x2 + w0 =0***

là nghiệm cần tìm.

Với điểm dữ liệu thuộc class 1 có nhãn là 1, điểm dữ liệu thuộc class 2 có có nhãn là -1. Ta có nhận xét các điểm nằm cùng 1 phía so với đường thẳng này sẽ làm cho hàm số ***fw (x)*** mang cùng dấu. Vậy nếu **w** là là nghiệm của bài toán Perceptron, với một điểm dữ liệu ***x*** chưa được gán nhãn, ta có thể xác định class của nó bằng phép toán đơn giản sau:

***Label(x) = 1 if wTx ≥ 0***, otherwise -1 hoặc ***label(x) =sgn(wTx)***

\* sgn là hàm xác định dấu với giả sử rằng sgn(0)=1.

## 2.1. Xây dựng hàm mất mát.

Khi thực hiện Classsify, chúng ta mong muốn rằng không có điểm nào bị misclassified (phân lớp lỗi), nên chúng ta cần tối thiểu hàm mất mát với hàm mất mát đơn giản nhất trong bài toán Perceptron là hàm đếm số lượng các điểm bị misclassified.

***J1(w)=∑(-yi .sgn(wTxi ))với xi ∈ M***

Trong đó ***M*** là tập hợp các điểm bị misclassified (thay dổi theo ***w***). Với mỗi điểm ***xi ∈ M, yi*** và ***sgn(wTx) khác nhau,*** do đó ***–yi .sgn(wTxi ) = 1***. Khi hàm số ***J1(w) = 0*** (đạt giá trị nhỏ nhất) thì ta không còn điểm nào bị misclassified.

\* Hàm số ***J1(w)*** là hàm rời rạc, không tính được đạo hàm theo w nên rất khó tối ưu.

Do đó ta cần tìm một hàm mất mất khác mà việc tối ưu khả thi hơn:

***J(w)=∑(-yi .wTxi )với xi ∈ M***

Từ hàm ***J1(w)*** ta bỏ đi hàm sgn, qua đó ta có một nhận xét mới rằng một điểm ***xi*** bị misclassified nằm càng xa boundary thì giá trị ***-yi .wTxi***  càng lớn (tức sự sai lệch càng lớn). Và giá trị nhỏ nhất của hàm ***J(w)=0*** nếu không có điểm nào bị misclassified.

Hàm ***J(w)*** được cho là tốt hơn hàm ***J1(W)*** vì nó “trừng phạt” rất nặng những điểm lấn sâu sang lãnh thổ của class khác, trong khi ***J1(w)*** “trừng phạt” các điểm bị misclassified như nhau (đều = 1) bất kể chúng gần hay cách xa boundary.

Tại một thời điểm nếu ta chỉ quan tâm tới các điểm bị misclassified thì hàm ***J(w)*** khả vi, nên chúng ta có thể sự dụng ***Gradient Descent*** hoặc ***Stochastis Gradient Descent (SGD)*** để tối ưu hàm mất mát này. Trong bài báo cáo này chúng ta sẽ sử dụng SGD.

Với một điểm dữ liệu ***xi*** misclassfied, hàm mất mát trở thành:

***J(w; xi ; yi ) = −yi wTxi***

Đạo hàm tương ứng:

***∇w J(w ; xi ;yi) = −yi xi***

Quy tắc cập nhật là:

***w = w + η.yi xi***

Với ***η*** là learning rate được chọn bằng 1, ta có:

***wt+1 = wt + yixi***

Tức với mỗi điểm ***xi*** bị classified, chỉ cần nhân điểm đó với nhãn ***yi***, lấy kết quả cộng vào ***w*** ta được ***w*** mới.

Ta có:

wt+1T. xi = (wt + yi xi)Txi = wiTxi + yi.||xi||22

Nếu ***yi = 1***, do ***xi*** misclassfied nên ***wTxi < 0***, vì ***yi = 1*** nên yi ||xi||22 >= 1 (***x0 = 1***), nghĩa là ***wt+1T xi > wTxi*** (do ***wt+1*** tiến về phía làm ***xi*** được phân lớp đúng). Tương tự khi ***y = -1***.

## 2.2. Chứng minh hội tụ.

Giả sử ***w\**** là nghiệm của PLA (với giả thuyết hai class là linearly separable), với mọi ***α > 0,*** nếu ***w\**** là nghiệm, ***αw\****  cũng là nghiệm của bài toán. Xét dãy số không âm ***uα(t) = ||wt – αw\*||22***  với ***xi*** là một điểm bị misclassified nếu dùng nghiệm ***wt*** ta có:

A math equations and symbols

Description automatically generated with medium confidence

Dấu nhỏ hơn ở dòng cuối là vì ***yi2= 1*** và ***2yi xiTwt < 0***, nếu ta đặt:

A group of math equations

Description automatically generated

Và chọn ***α=β2/γ***, ta có:

A group of symbols with a plus and a plus

Description automatically generated with medium confidence

Công thức nói lên rằng nếu luôn có điểm bị misclassified thì dãy ***uα(t)*** là dãy giảm, bị chặn dưới bởi 0, và phần tử sau kém phần tử trước ít nhất một lượng là ***β2>0 (vô lý, do đó PLA sẽ hội tụ sau một số hữu hạn bước).***

## 2.3. Tóm tắt PLA và một vài lưu ý.

Các bước của PLA:

Bước 1: chọn ngẫu nhiên một vector hệ số ***w*** với các phần tử gần 0.

Bước 2: Duyệt ngẫu nhiên qua từng điểm dữ liệu ***xi***:

* Nếu ***xi*** được phân lớp đúng, tức ***sgn(wTxi) = yi*** thì không cần làm gì.
* Ngược lại, nếu ***xi*** misclassified, cập nhật ***w*** theo công thức:

***W = w + yi xi***

Bước 3: Kiểm tra xem có bao nhiêu điểm bị misclassified, nếu còn quay lại bước 2, nếu không thì dừng thuật toán.

Một vài điểm lưu ý:

* PLA đòi hỏi dữ liệu linearly separable, do đó nếu có nhiễu thì PLA sẽ không hội tụ được.
* PLA có thể cho ra vô số nghiệm khác nhau: vì nếu hai class là linearly separable thì có vô số đường thẳng phân cách 2 class đó.
* Cải tiến nhỏ cho PLA: Pocket Algorithm – tìm boundary cho hai class sao cho ít điểm bị misclassified nhất khi có nhiễu.

---------------------------------------KẾT THÚC---------------------------------------