Vanishing/Eploding Gradient Problem

Thuật toán Backpropagation

Một mô hình neural-network trải qua ba giai đoạn:

- Feed- forward : chuyển tiếp từ input thành output
- Backpropagation: Tính gradient của hàm loss từ đầu ra ngược lại đến đầu vào
- Update các trọng số

Lưu ý: Backpropagation đơn giản chỉ là tính đạo hàm của hàm hợp

Ví dụ : input = [x,w1,b1] với w,b là trọng số (đc khởi tạo ngẫu nhiên cần update)

-> h1 =sigmoid(w1x + b1)

-> h2 = sigmoid(w2h1+b2)

L = Loss(h2,y)

-> *dL/dw1 = dL/dh2 * dh2/dh1 * dh1/dw1* → ***update**: w1 = w1 + dL/dw1 (1)*

….. tương tự với b1,w2,b2

Thế nào là Vanishing/Exploding Gradient

Hiện tượng trong quá trình backpropgation thì gradient bị quá nhỏ hoặc quá lớn ảnh hưởng đến quá trình cập nhật trọng số

- Vanishing

  Ta có:sigmoid'(x) = sigmoid(x) * (1-sigmoid(x))

  Nếu x = 5 → sigmoid'(x) = 0.0006 -> rất nhỏ

  Và có thể thấy khi mô hình càng sâu thì (1) càng dài càng có nhiều phép nhân kiểu như 0.1^100.. từ đó khiến dL/dw1 rất rất nhỏ và làm cho w1 ko update được gì cả và không bao giờ hội tụ đến tối ưu → đây được gọi là Vanishing Gradient
- Exploding

  Ngược lại với Vanishing là Exploding khi gradient ngày càng lớn và khiến quá trình không hội tụ mà càng ngày càng phân kỳ

Dấu hiệu bị Vanishing :

- Trọng số của các lớp cuối thay đổi nhưng của các lớp đầu gần như 0 thay đổi
- Mô hình học rất chậm, giá trị loss gần như ko đổi

Dấu hiệu bị Exploding:

- Trọng số mô hình thay đổi nhanh theo cấp số nhân thậm chí bị tràn và trả về Nan
- Giá trị loss dao động mạnh, ko ổn định

Giải pháp

Weight Initialization

Khởi tạo trọng số là một bước quan trọng trong quá trình xây dựng mô hình. Phương pháp khởi tạo trọng số có thể giúp mô hình được huấn luyện dễ dàng, giảm thiểu sự phụ thuộc vào learning rate và tăng độ chính xác của mô hình.

Có thể thấy hiện tượng vanishing/exploding xảy ra khi trọng số quá lớn hoặc quá nhỏ, khi đó phương pháp khởi tạo trọng số giúp điều chỉnh độ lớn của trọng số

Có một số cách khởi tạo trọng số:

1.Random Normal Initialization

- Các giá trị trọng số được lấy ngẫu nhiên từ phân phối chuẩn với mean = 0 và std = 1/sqrt(n)

Với n là số lượng neuron đầu vào của lớp đó → khi đầu vào càng lớn(càng nhiều neuron thì trọng số càng nhỏ)

2.Xavier Initialization

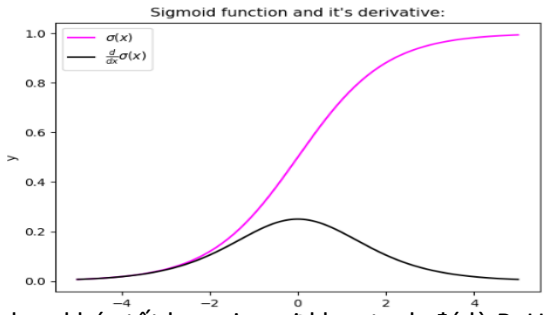
Cũng như cách khởi tạo trên trọng số được lấy ngẫu nhiên với phân phối chuẩn với mean = 0 và std = 1/sqrt(n) nhưng n ở đây bằng tổng neuron đầu vào và đầu ra của lớp đó. Giúp trọng số phù hợp với kích thước của lớp

3.He Initialization

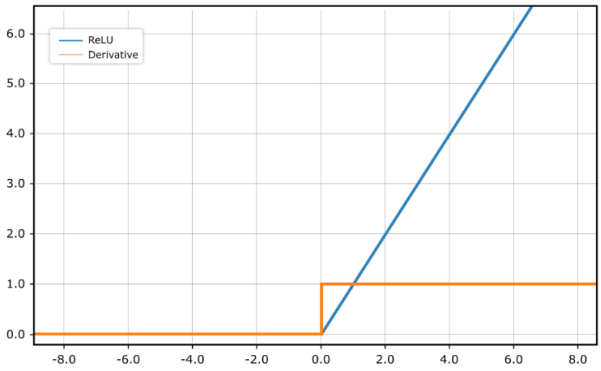
Phương pháp này tương tự với phương pháp 2 nhưng std = 1/sqrt(n/2) trong n là tổng đầu vào của lớp đó. Phương pháp này phù hợp với các activation như ReLU

Nonsaturating Activation Functions

Chúng ta có thể dễ dàng thấy được vấn đề vanishing/exploding cũng một phần do sự lựa chọn các activation. Ví dụ như hàm sigmoid : rõ ràng với các giá trị âm và dương (hơn lớn 1 chút) thì gradient của nó đã biến mất (~ 0)

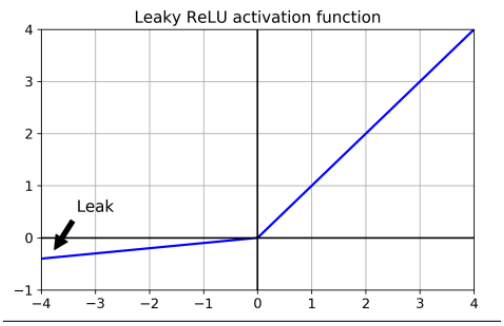


Chúng ta có một vài lựa chọn khác tốt hơn sigmoid hay tanh đó là ReLU ..chủ yếu vì nó ko bị "saturate" với các giá trị dương như sigmoid



Thật không may thì hàm ReLU cũng không hoàn hảo. Hiện tượng Dying ReLU là khi mô hình của bản luôn xuất ra cùng 1 giá trị là 0 cho bất kỳ đầu vào nào. Điều này xảy ra trọng số của nó rơi vào các giá trị âm. Và hàm ReLU thì không có khả năng tự phục hồi và thoát khỏi trạng thái này(khác với sigmoid và tanh khi 2 hàm này có độ cong nhất định và có thể phục hồi sau khoảng thời gian dài)

Từ đó có ra đời các biến thể của ReLU là Leaky ReLU hay ELU



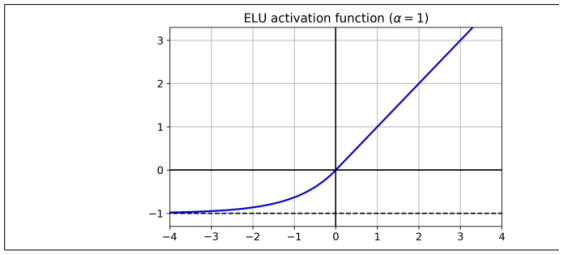


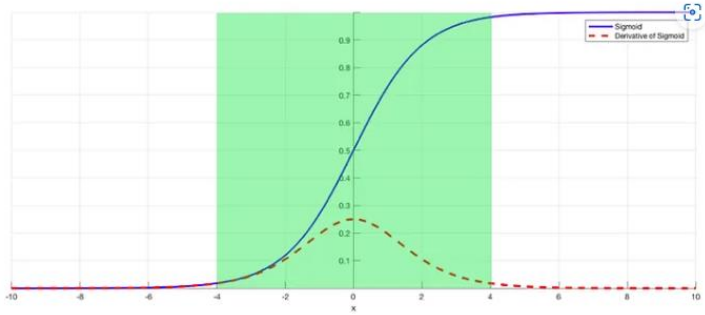
Figure 11-3. ELU activation function

Batch Normalization

Batch Normalization là một phương pháp chuẩn hóa các giá trị đầu vào của một lớp theo một phân phối chuẩn – bằng cách tính mean và std của trọng số sau đó trừ đi mean rồi chia cho std.

BN có thể được hình dung đơn giản như một lớp bổ sung trong mạng giúp chuẩn hóa dữ liệu trước khi đưa nó vào các activation function.

Nhớ lại tại sao hàm sigmoid và tanh dễ bị vanishing ….là do nó có các khoảng bão hòa (các khoảng âm và dương hơi lớn 1 chút) , vậy chỉ cần đảm bảo trọng số nằm trong " phạm vi tốt"(màu xanh) và không chạm tới "vùng bão hòa" của hàm sigmoid và tanh thì có thể hạn chế được vanishing. Và BN có thể làm được việc này khi đặt nó ở mỗi lớp



Tuy nhiên chỉ mình BN thì cũng không thể ngăn được Vanishing với hàm sigmoid hay tanh do kể cả trọng số nằm trong khoảng xanh thì đạo hàm của nó vẫn < 1 và khi mô hình quá sâu thì vẫn làm cho đạo hàm ~ 0.

Gradient Clipping

Gradient Clipping là phương pháp giới hạn gradient trong một khoảng cho trước, để tránh gradient quá lớn hoặc quá nhỏ và gây ra vanishing/exploding. Khi giá trị gradient vượt ngưỡng thiết lập, nó sẽ được cắt giảm để nó ko vượt quá giới hạn. Phương pháp này được áp dụng sau khi tính gradient và trước khi cập nhật trọng số.

Phương pháp này thường áp dụng cho mạng RNN vì BN khá khó dùng trong RNN