

# VIP Cheatsheet: Học sâu

Afshine AMIDI và Shervine AMIDI

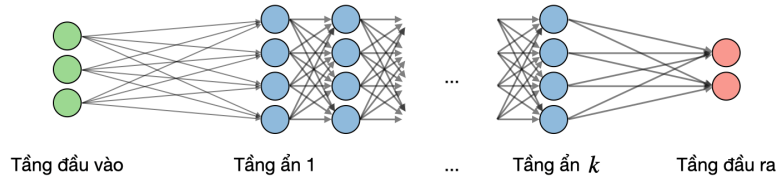
Ngày 17 tháng 5 năm 2020

Dịch bởi Trần Tuấn Anh, Phạm Hồng Vinh, Đàm Minh Tiến, Nguyễn Khánh Hưng, Hoàng Vũ Đạt và Nguyễn Trí Minh

## Mạng Neural

Mạng Neural là 1 lớp của các mô hình (models) được xây dựng với các tầng (layers). Các loại mạng Neural thường được sử dụng bao gồm: Mạng Neural tích chập (Convolutional Neural Networks) và Mạng Neural hồi quy (Recurrent Neural Networks).

□ **Kiến trúc** – Các thuật ngữ xoay quanh kiến trúc của mạng neural được mô tả như hình phía dưới:



Bằng việc kí hiệu  $i$  là tầng thứ  $i$  của mạng,  $j$  là hidden unit (đơn vị ẩn) thứ  $j$  của tầng, ta có:

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

chúng ta kí hiệu  $w, b, z$  tương ứng với trọng số (weights), bias và đầu ra.

□ **Hàm kích hoạt (Activation function)** – Hàm kích hoạt được sử dụng ở phần cuối của đơn vị ẩn để đưa ra độ phức tạp phi tuyến tính (non-linear) cho mô hình (model). Đây là những trường hợp phổ biến nhất:

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$

□ **Lỗi (loss) Cross-entropy** – Trong bối cảnh của mạng neural, hàm lỗi cross-entropy  $L(z, y)$  thường được sử dụng và định nghĩa như sau:

$$L(z, y) = - \left[ y \log(z) + (1 - y) \log(1 - z) \right]$$

□ **Tốc độ học (Learning rate)** – Tốc độ học, thường được kí hiệu bởi  $\alpha$  hoặc đôi khi là  $\eta$ , chỉ ra tốc độ mà trọng số được cập nhật. Thông số này có thể là cố định hoặc được thay đổi tùy biến. Phương thức (method) phổ biến nhất hiện tại là Adam, đó là phương thức thay đổi tốc độ học một cách phù hợp nhất có thể.

□ **Backpropagation (Lan truyền ngược)** – Backpropagation là phương thức dùng để cập nhật trọng số trong mạng neural bằng cách tính toán đầu ra thực sự và đầu ra mong muốn. Đạo hàm theo trọng số  $w$  được tính bằng cách sử dụng quy tắc chuỗi (chain rule) theo như cách dưới đây:

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Như kết quả, trọng số được cập nhật như sau:

$$w \leftarrow w - \eta \frac{\partial L(z, y)}{\partial w}$$

□ **Cập nhật trọng số** – Trong mạng neural, trọng số được cập nhật như sau:

- **Bước 1:** Lấy một mẻ (batch) dữ liệu huấn luyện (training data).
- **Bước 2:** Thực thi lan truyền tiến (forward propagation) để lấy được lỗi (loss) tương ứng.
- **Bước 3:** Lan truyền ngược lỗi để lấy được gradients (độ dốc).
- **Bước 4:** Sử dụng gradients để cập nhật trọng số của mạng (network).

□ **Dropout** – Dropout là thuật ngữ kĩ thuật dùng trong việc tránh overfitting tập dữ liệu huấn luyện bằng việc bỏ đi các đơn vị trong mạng neural. Trong thực tế, các neurals hoặc là bị bỏ đi bởi xác suất  $p$  hoặc được giữ lại với xác suất  $1 - p$

## Mạng neural tích chập (Convolutional Neural Networks)

□ **Yêu cầu của tầng tích chập (Convolutional layer)** – Bằng việc ghi chú  $W$  là kích cỡ của volume đầu vào,  $F$  là kích cỡ của neurals thuộc convolutional layer,  $P$  là số lượng zero padding, khi đó số lượng neurals  $N$  phù hợp với volume cho trước sẽ như sau:

$$N = \frac{W - F + 2P}{S} + 1$$

□ **Batch normalization (chuẩn hoá)** – Đây là bước mà các hyperparameter  $\gamma, \beta$  chuẩn hoá batch  $\{x_i\}$ . Bằng việc kí hiệu  $\mu_B, \sigma_B^2$  là giá trị trung bình, phương sai mà ta muốn gán cho batch, nó được thực hiện như sau:

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Nó thường được tính sau fully connected/convolutional layer và trước non-linearity layer và mục tiêu là cho phép tốc độ học cao hơn cũng như giảm đi sự phụ thuộc mạnh mẽ vào việc khởi tạo.

## Mạng neural hồi quy (Recurrent Neural Networks)

❑ **Các loại cổng** – Đây là các loại cổng (gate) khác nhau mà chúng ta sẽ gặp ở một mạng neural hồi quy điển hình:

Cổng đầu vào	cổng quên	cổng đầu ra	cổng
Ghi vào cell hay không?	Xoá cell hay không?	Cần tiết lộ bao nhiêu về cell?	Ghi bao nhiêu vào cell?

❑ **LSTM** – Mạng bộ nhớ dài-ngắn (LSTM) là 1 loại RNN model tránh vấn đề vanishing gradient (gradient biến mất đột ngột) bằng cách thêm vào cổng 'quên' ('forget' gates).

## Học tăng cường (Reinforcement Learning) và điều khiển

Mục tiêu của học tăng cường đó là cho tác tử (agent) học cách làm sao để tối ưu hoá trong một môi trường.

❑ **Tiến trình quyết định Markov (Markov decision processes)** – Tiến trình quyết định Markov (MDP) là một dạng 5-tuple  $(\mathcal{S}, \mathcal{A}, \{P_{sa}\}, \gamma, R)$  mà ở đó:

- $\mathcal{S}$  là tập hợp các trạng thái (states)
- $\mathcal{A}$  là tập hợp các hành động (actions)
- $\{P_{sa}\}$  là xác suất chuyển tiếp trạng thái cho  $s \in \mathcal{S}$  và  $a \in \mathcal{A}$
- $\gamma \in [0, 1]$  là discount factor
- $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  hoặc  $R: \mathcal{S} \rightarrow \mathbb{R}$  là reward function (hàm định nghĩa phần thưởng) mà giải thuật muốn tối đa hoá

❑ **Policy** – Policy  $\pi$  là 1 hàm  $\pi: \mathcal{S} \rightarrow \mathcal{A}$  có nhiệm vụ ánh xạ states tới actions.

*Chú ý: Ta quy ước rằng ta thực thi policy  $\pi$  cho trước nếu cho trước state  $s$  ta có action  $a = \pi(s)$ .*

❑ **Hàm giá trị (Value function)** – Với policy cho trước  $\pi$  và state  $s$ , ta định nghĩa value function  $V^\pi$  như sau:

$$V^\pi(s) = E \left[ R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi \right]$$

❑ **Phương trình Bellman** – Phương trình tối ưu Bellman đặc trưng hoá value function  $V^{\pi^*}$  của policy tối ưu (optimal policy)  $\pi^*$ :

$$V^{\pi^*}(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in \mathcal{S}} P_{sa}(s') V^{\pi^*}(s')$$

*Chú ý: ta quy ước optimal policy  $\pi^*$  đối với state  $s$  cho trước như sau:*

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{sa}(s') V^*(s')$$

❑ **Giải thuật duyệt giá trị (Value iteration)** – Giải thuật duyệt giá trị gồm 2 bước:

- 1) Ta khởi tạo giá trị:

$$V_0(s) = 0$$

- 2) Ta duyệt qua giá trị dựa theo giá trị phía trước:

$$V_{i+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left[ \sum_{s' \in \mathcal{S}} \gamma P_{sa}(s') V_i(s') \right]$$

❑ **Ước lượng khả năng tối đa (Maximum likelihood estimate)** – Ước lượng khả năng tối đa cho xác suất chuyển tiếp trạng thái (state) sẽ như sau:

$$P_{sa}(s') = \frac{\# \text{thời gian hành động } a \text{ tiêu tốn cho state } s \text{ và biến đổi nó thành } s'}{\# \text{thời gian hành động } a \text{ tiêu tốn cho state (trạng thái) } s}$$

❑ **Q-learning** – Q-learning là 1 dạng phán đoán phi mô hình (model-free) của Q, được thực hiện như sau:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$