# LÝ THUYẾT COMPUTATIONAL GRAPH

Mục lục

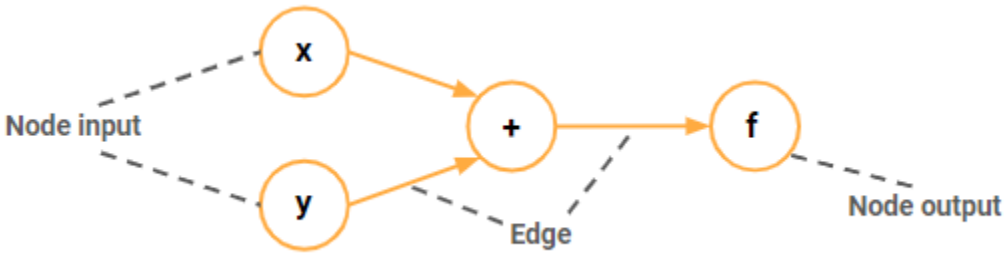


## 1. Computational Graph

- Computational Graph (Đồ thị tính toán) là một biểu đồ biểu diễn mối quan hệ giữa các phép toán và dữ liệu trong một mô hình tính toán.
- Computational Graph là đồ thị có hướng.

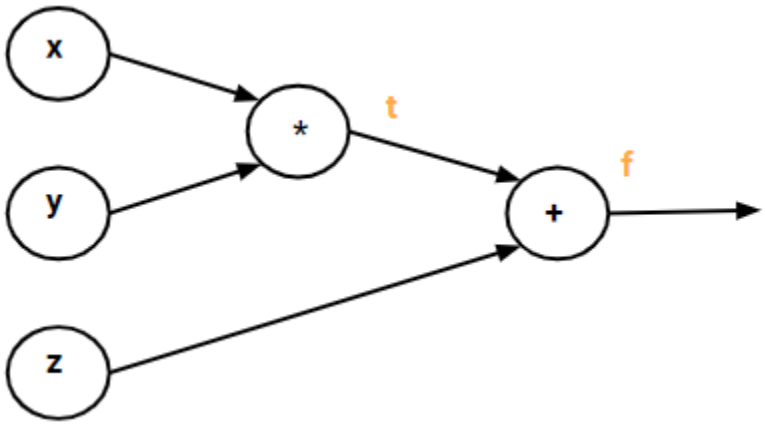
### a) Đặc điểm

- Nodes (nút): đại diện cho các phép toán hoặc biến trong mô hình. Node có thể là một ma trận, một vector hoặc scalar (mình không dịch được, nôm na gọi là giá trị hay con số).
- Edges (cạnh): có thể chứa thông tin về trọng số hoặc các thông số liên quan đến quá trình tính toán.

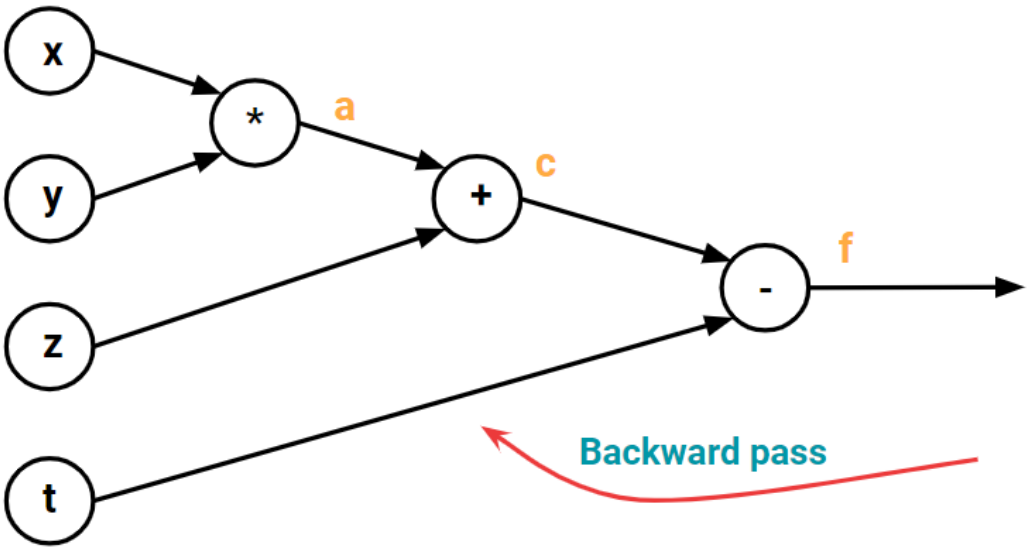



### b) Thuật toán

- Thuật toán forward pass (lan truyền thẳng, mình tạm dịch là vậy) và thuật toán backward pass (lan truyền ngược). Mình sẽ lấy ví dụ trong slide để các bạn dễ hình dung hơn.
- Đầu tiên, xây dựng 1 đồ thị tính toán f đơn giản là f(x, y, z) = (x * y) + z




- Đặt t = x * y, lúc này hàm f sẽ còn 2 biến là f(t, z) = t + z
- Nếu cho các giá trị của x, y, z lần lượt là 2, 3, 4 thì ta sẽ tính được f = 10, đây là kết quả đầu ra.
- Thuật toán forward pass chỉ giúp ta tính giá trị đầu ra thông qua các phép toán như vậy.
- Tiếp theo, thuật toán backward pass (hay còn được gọi là Backpropagation, lan truyền ngược).
- *Backpropagation sẽ giúp chúng ta tính đạo hàm dễ dàng hơn*.
- Vậy dễ dàng hơn như thế nào?
- Mình xin phép lấy tiếp ví dụ trong slide với một phép toán phức tạp hơn chút.




- Trong đồ thị tính toán trên, ta có 4 input (x, y, z, t).
- Đạo hàm $\frac{df}{dz}$ và $\frac{df}{dt}$ khá dễ dàng tính được vì nó chỉ là phép cộng và phép trừ thôi.

- Còn đạo hàm f theo biến x và y thì sẽ tính theo *quy tắc dây chuyền* (chain rule)

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\frac{\partial a}{\partial x} = 1.1.y = y$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\frac{\partial a}{\partial y} = 1.1.x = x$$

- Ta nhận thấy rằng, khi tính đạo hàm của hàm f theo biến x và y thì có cùng $\frac{df}{dc} * \frac{dc}{da}$ , Computational Graph giúp tái sử dụng các giá trị này, từ đó tiết kiệm được thời gian và chi phí tính toán.
- Nãy giờ, mình đã trình bày lý thuyết về Computational Graph.
- Mình sẽ làm rõ hơn về việc thuật toán backward pass (backpropagation) tính đạo hàm để làm gì.

- Câu trả lời là dùng để tính đạo hàm của hàm mất mát (loss function) với đầu vào ở đây là các trọng số (weight) và độ thiên lệch (bias).

- Để cập nhật trọng số và bias, ta sẽ sử dụng thuật toán Gradient Descent.
- Các bạn có thể tham khảo thêm về thuật toán Gradient Descent *tại đây*

## 2. Ưu và nhược điểm (phần này mình hỏi chatGPT 😕)

### a) Ưu điểm

Tính Hiển Thị Cấu Trúc:

- Đồ thị tính toán cung cấp một cách hiển thị rõ ràng và trực quan về cấu trúc của mô hình. Nó giúp người lập trình và nhà nghiên cứu dễ dàng theo dõi và hiểu quá trình tính toán.

Lan Truyền Ngược Hiệu Quả:

- Đồ thị tính toán làm cho quá trình lan truyền ngược trở nên hiệu quả. Bằng cách tính toán đạo hàm theo chiều ngược, mô hình có thể được cập nhật một cách hiệu quả qua nhiều lớp.

Tối Ưu Hóa:

- Các framework deep learning như TensorFlow và PyTorch sử dụng đồ thị tính toán để tối ưu hóa và tận dụng tài nguyên máy tính hiệu quả. Điều này bao gồm tối ưu hóa bộ nhớ và khả năng song song của GPU.

Tính Dễ Diễn Giải:

- Đồ thị tính toán giúp trong việc diễn giải và giải thích mô hình. Bạn có thể xem trực tiếp cấu trúc tính toán và xác định các phép toán quan trọng.

### b) Nhược điểm

Phức tạp khi lập trình

- Đôi khi, việc xây dựng và quản lý đồ thị tính toán có thể làm tăng sự phức tạp trong quá trình lập trình. Điều này đặc biệt đúng khi mô hình có cấu trúc phức tạp và chứa nhiều lớp.

Overhead Tính Toán:

- Trong một số trường hợp, có thể có overhead tính toán do việc quản lý đồ thị. Điều này có thể dẫn đến việc tăng thời gian chạy và tài nguyên tiêu tốn.

Khó Khăn trong Debugging:

- Việc debug trong môi trường đồ thị tính toán có thể khó khăn hơn so với việc debug trong các mô hình thông thường. Hiểu rõ về cách các phép toán lan truyền và ngược lại có thể đòi hỏi kiến thức sâu về mô hình.

Khả Năng Hiểu Quả Phụ Thuộc vào Cấu Trúc:

- Hiệu suất của việc sử dụng đồ thị tính toán phụ thuộc nhiều vào cấu trúc của mô hình và cách triển khai của framework sử dụng.

## 3. Ứng dụng

- Là công cụ mạnh để đạo hàm theo từng biến.
- Trong các bài toán network nhiều lớp, cần phải tính đạo hàm theo từng biến. Để làm việc này hiệu quả hơn, người ta sử dụng thuật toán lan truyền ngược (backpropagation).
- Parallelization (tính toán song song)

4. Nguồn tham khảo

https://www.geeksforgeeks.org/computational-graphs-in-deep-learning/

https://www.youtube.com/watch?v=zIBhOKXQOXU

https://github.com/lmnguyet/CS523-Computational-Graph