Midterm Project

Deep Learning course

**Question 1 (5 points):**

Understanding and Implementing automatic differentiation (Autograd) in Deep Learning in the three following aspects:

1. Theoretical Component:

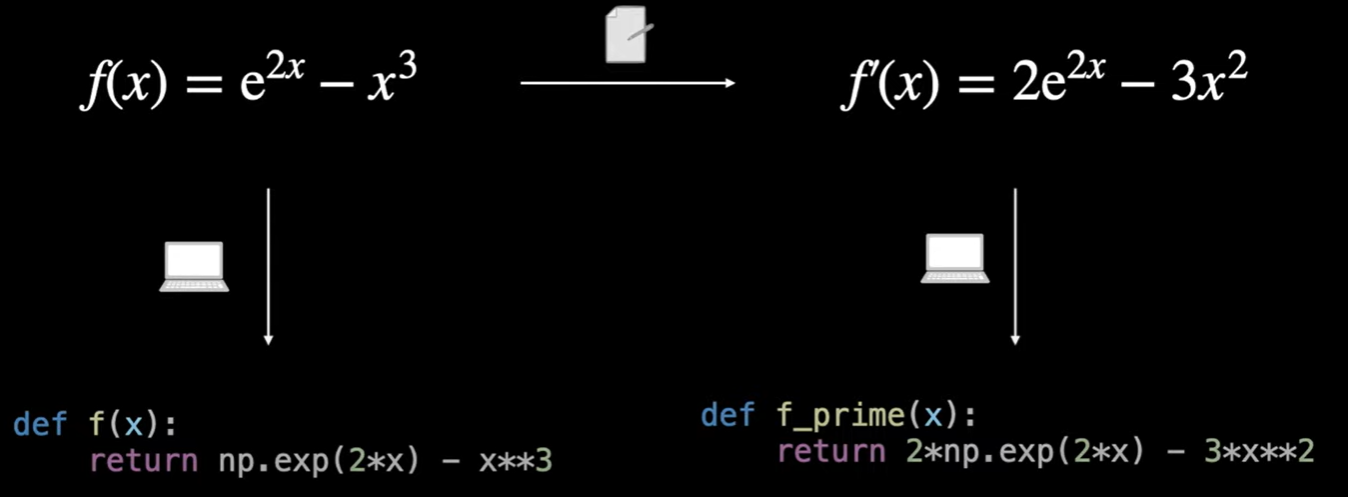
* Provide a detailed explanation of automatic differentiation (AD) and its significance in training deep learning models.
* Compare and contrast forward mode and reverse mode automatic differentiation, discussing their advantages and disadvantages.
* Explain the role of computational graphs in automatic differentiation and backpropagation.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

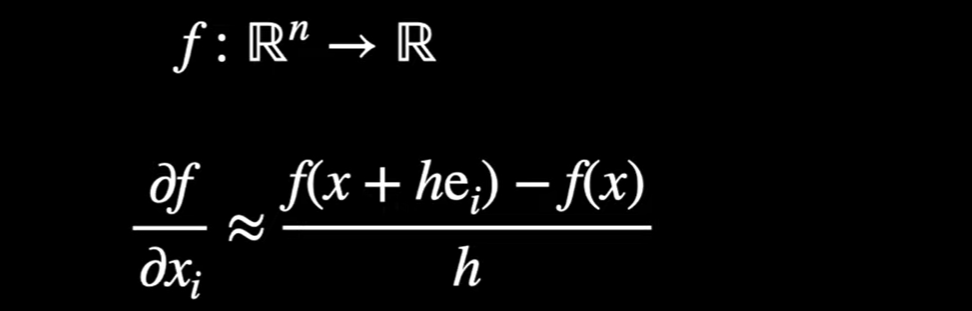
1. Automatic differentiation (AD)

Các phương pháp khác nhau để tính toán đạo hàm của một hàm số:

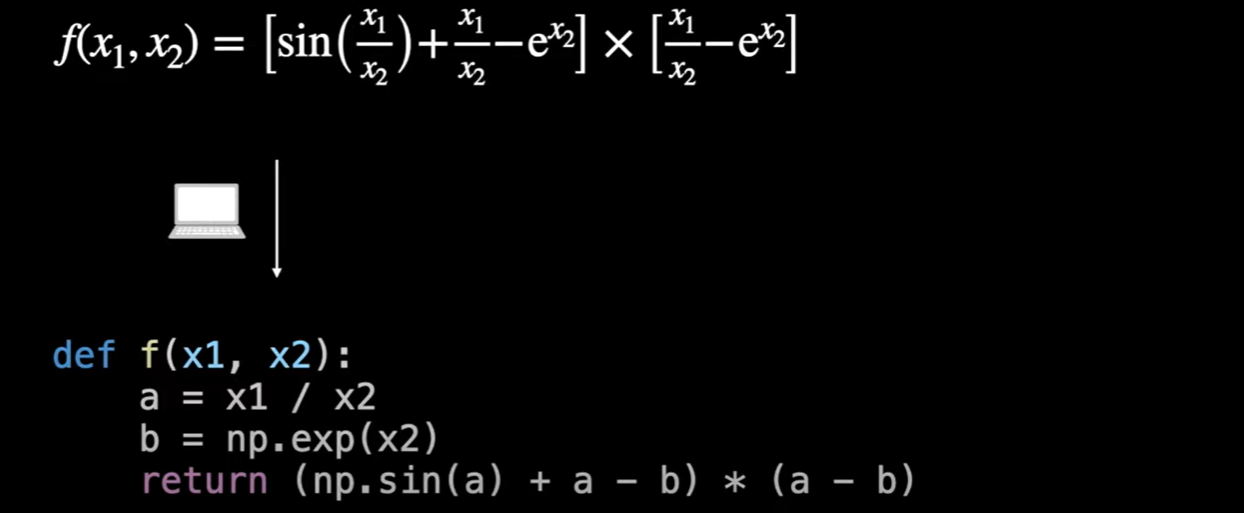
+ Manual: Tính toán đạo hàm bằng cách thủ công, tức là sử dụng quy tắc đạo hàm của các hàm cơ bản và áp dụng chúng theo từng bước cho hàm cần tính đạo hàm.



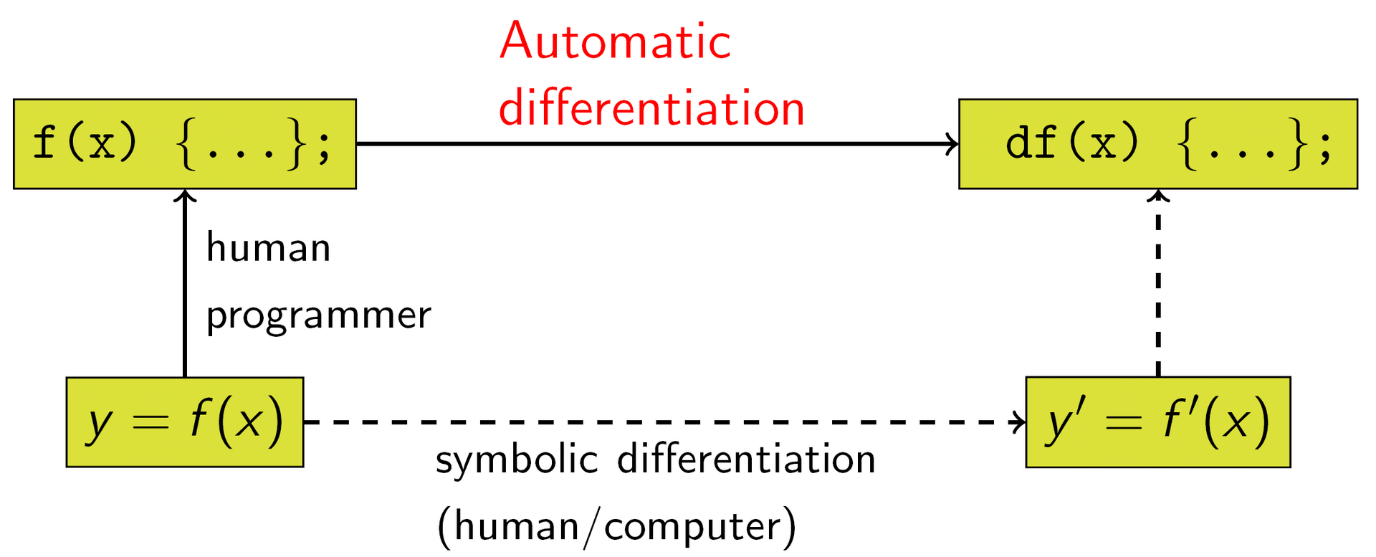
+ Numerical: Tính toán đạo hàm bằng cách xấp xỉ bằng giới hạn, như sử dụng phương pháp sai phân hữu hạn để ước lượng giá trị của đạo hàm tại một điểm bằng cách sử dụng các giá trị của hàm tại các điểm gần nhau.



+ Symbolic: Tính toán đạo hàm bằng cách sử dụng biểu thức ký hiệu của hàm để tìm ra biểu thức ký hiệu của đạo hàm, tức là không cần phải tính toán giá trị cụ thể của hàm mà chỉ cần biểu thị dưới dạng biểu thức toán học.



+ Automatic: Tính toán đạo hàm bằng cách sử dụng các thuật toán tự động tính toán đạo hàm, mà không cần sự can thiệp thủ công. Phương pháp này còn được gọi là "tích phân tự động", trong đó AD là viết tắt của "Automatic Differentiation".



Cả bốn phương pháp này đều có ưu và nhược điểm riêng, và sự lựa chọn giữa chúng phụ thuộc vào ngữ cảnh cụ thể của bài toán và yêu cầu về độ chính xác và hiệu suất tính toán. Trong học sâu và các ứng dụng liên quan đến gradient, Automatic Differentiation thường được ưa chuộng vì nó kết hợp được sự hiệu quả và chính xác cao.

Định nghĩa:

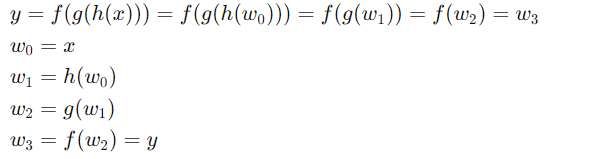
Automatic differentiation (AD), cho phép tính toán hiệu quả và chính xác các đạo hàm của các hàm, bao gồm các sự kết hợp phức tạp của các hàm, liên quan đến các biến đầu vào của chúng.

AD đóng vai trò quan trọng trong việc huấn luyện các mô hình học sâu vì nó cho phép các thuật toán tối ưu hóa dựa trên đạo hàm, như gradient descent ngẫu nhiên (SGD) và các biến thể của nó, tìm ra các tham số tối ưu của mô hình một cách hiệu quả thông qua việc cập nhật chúng dựa trên các đạo hàm của một hàm mất mát liên quan đến các tham số đó.

Tích phân tự động khai thác sự thực rằng mọi tính toán trên máy tính, dù phức tạp đến đâu, đều thực hiện một chuỗi các phép toán số học cơ bản (cộng, trừ, nhân, chia, v.v.) và các hàm toán cơ bản (exp, log, sin, cos, v.v.). Bằng cách áp dụng quy tắc chuỗi lặp đi lặp lại cho các phép toán này, các đạo hàm riêng của bất kỳ cấp bậc nào cũng có thể được tính tự động, chính xác đến độ chính xác làm việc, và sử dụng tối đa một hằng số nhỏ hơn số lượng phép toán hơn so với chương trình gốc.

* Đây là một trong những nguyên nhân chính sau thành công của việc huấn luyện các mạng nơ-ron sâu với nhiều tầng ẩn. Nó cũng đóng vai trò quan trọng trong các phương pháp thống kê Bayesian hiện đại sử dụng thông tin đạo hàm để học. Lập trình có thể vi phân là một thuật ngữ mới đây để chỉ các kỹ thuật và trường hợp sử dụng của AD.

Tích phân tự động cơ bản dựa trên việc phân rã các khác biệt được cung cấp bởi quy tắc chuỗi của các đạo hàm riêng của các hàm hợp. Đối với sự kết hợp đơn giản:



quy tắc chuỗi cho:



Trong đó:

* *y* là hàm cuối cùng của chuỗi, đại diện cho kết quả cuối cùng của quá trình tính toán.
* *x* là biến đầu vào ban đầu.
* *w*0​,*w*1​,*w*2​,*w*3​ là các giá trị trung gian, kết quả từ các bước trung gian của quá trình tính toán.
* *h*, *g*, *f* là các hàm được kết hợp với nhau, tạo thành một chuỗi hàm.

*Có 2 loại* Automatic differentiation:

+ Tích lũy tiến (forward accumulation mode)

Thường được gọi là cách tiếp cận từ dưới lên (bottom-up), chế độ tiến (forward mode) hoặc chế độ tiếp xúc (tangent mode). Trong tích lũy tiến, ta duyệt qua quy tắc chuỗi từ bên trong ra bên ngoài.

+ Tích lũy ngược (reverse accumulation mode)

Thường được gọi là cách tiếp cận từ trên xuống (top-down), chế độ ngược (reverse mode) hoặc chế độ phản hồi (adjoint mode). Trong tích lũy ngược, ta duyệt qua quy tắc chuỗi từ bên ngoài vào bên trong.

Ý nghĩa của AD trong việc huấn luyện các mô hình deep learning:

* Tính toán Gradient hiệu quả: các mô hình học sâu thường liên quan đến các sự kết hợp phức tạp của các hàm với hàng triệu tham số. AD cho phép tính toán gradient của hàm mất mát liên quan đến các tham số này một cách hiệu quả, điều này rất quan trọng cho việc huấn luyện mô hình bằng các thuật toán tối ưu hóa dựa trên gradient.
* Thuật toán Backpropagation: thuật toán chính được sử dụng để huấn luyện các mô hình học sâu, và nó phụ thuộc vào AD để tính toán gradient một cách hiệu quả. Trong backpropagation, gradient được truyền ngược qua biểu đồ tính toán của mô hình để cập nhật các tham số của mô hình bằng cách sử dụng gradient descent.
* Tính linh hoạt và sự diễn đạt: AD cho phép các chuyên gia học sâu xác định các mô hình phức tạp với kiến trúc và các hàm kích hoạt tùy ý mà không cần phải tự mình tính toán và triển khai các tính toán gradient tương ứng. Sự linh hoạt này tăng tốc quá trình thí nghiệm và phát triển trong nghiên cứu và ứng dụng học sâu.
* Hỗ trợ tích hợp: AD được tích hợp vào các thư viện và framework phổ biến như TensorFlow, PyTorch, giúp cho việc lập trình và triển khai mô hình trở nên dễ dàng và linh hoạt hơn.

1. So sánh và phân biệt forward mode và reverse mode:

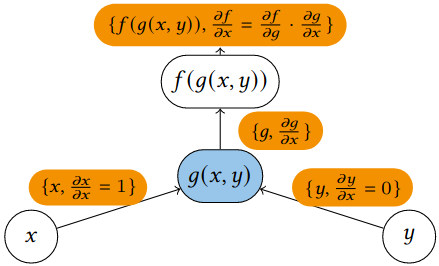
Forward accumulation (còn gọi là bottom-up, forward mode, tangent mode)

Reverse accumulation (còn gọi là top-down, reverse mode, or adjoint mode)

* 1. Forward mode

Forward accumulation thì ta duyệt qua quy tắc chuỗi từ bên trong ra bên ngoài.

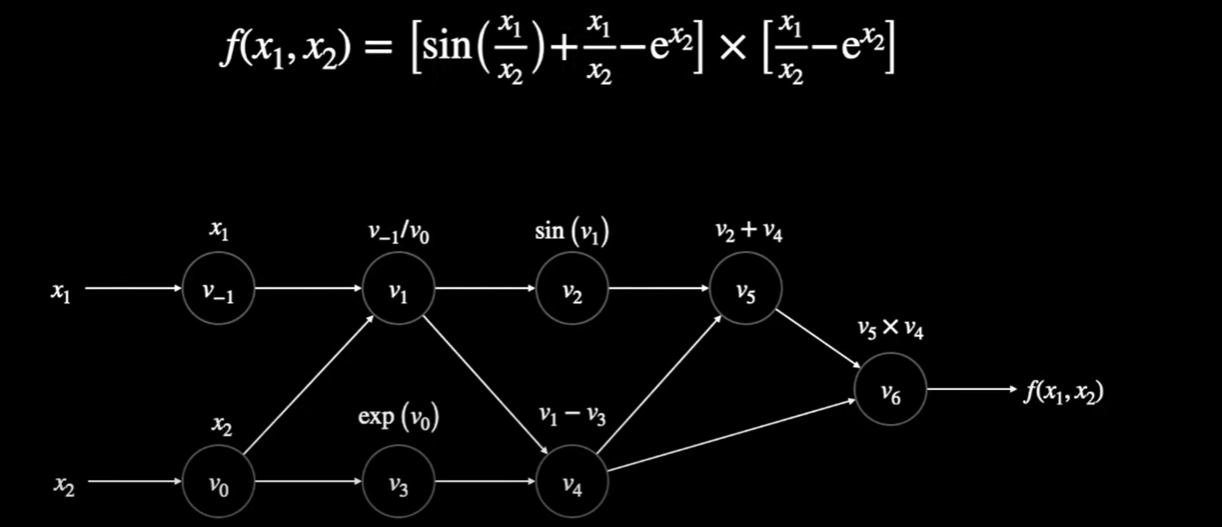
Trước tiên ta cố định biến độc lập mà theo đó sẽ tiến hành việc tính đạo hàm, sau đó tính đạo hàm của mỗi biểu thức con một cách đệ quy. Trong tính toán bằng bút và giấy, điều này liên quan đến việc lặp đi lặp lại việc thay thế đạo hàm của các hàm bên trong trong quy tắc chuỗi:

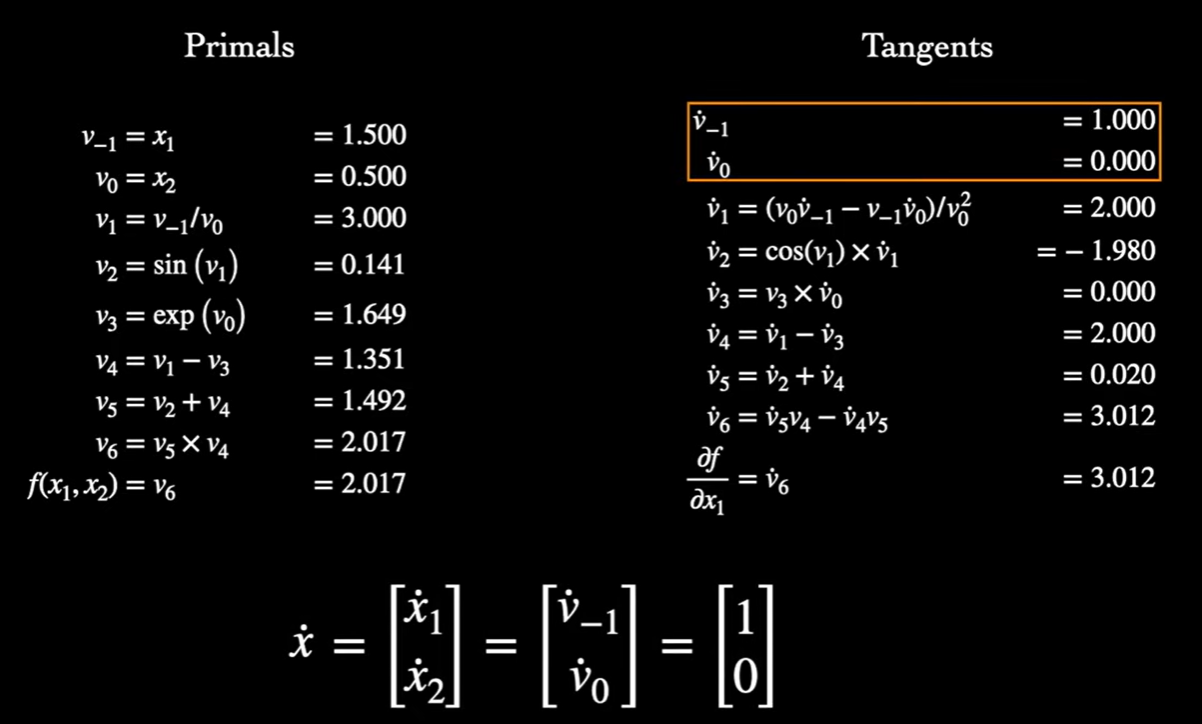


Đầu tiên sau đó  ∂�2/∂�1 vaà c  và cuối cùng ∂�/∂�2

Forward mode tính đệ quy:







Ưu điểm:

Phù hợp khi số lượng biến đầu vào (độ dài của vector đầu vào) ít hơn số lượng biến đầu ra (độ dài của vector đạo hàm), do đó thích hợp cho các mô hình có kích thước lớn.

Tính toán đạo hàm từng phần tử một, giúp tiết kiệm bộ nhớ trong trường hợp số lượng biến đầu ra lớn.

Nhược điểm:

Yêu cầu số lượng lớn các phép tính để tính toán đạo hàm đầy đủ, điều này có thể dẫn đến hiệu suất tính toán kém.

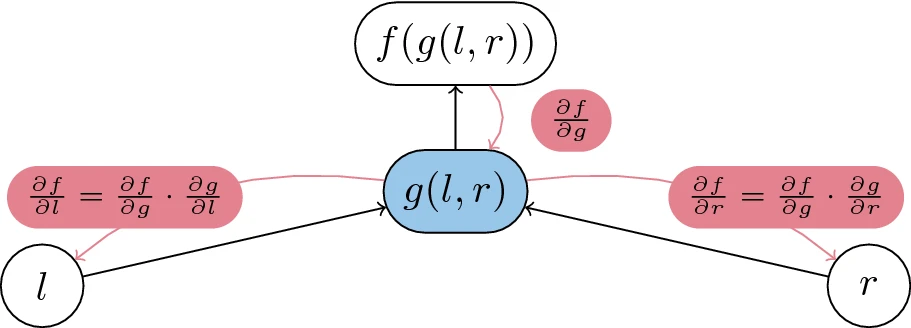
Không phù hợp cho các mô hình có số lượng biến đầu vào lớn.

* 1. Reverse mode

Forward accumulation thì ta duyệt qua quy tắc chuỗi từ bên ngoài vào bên trong.

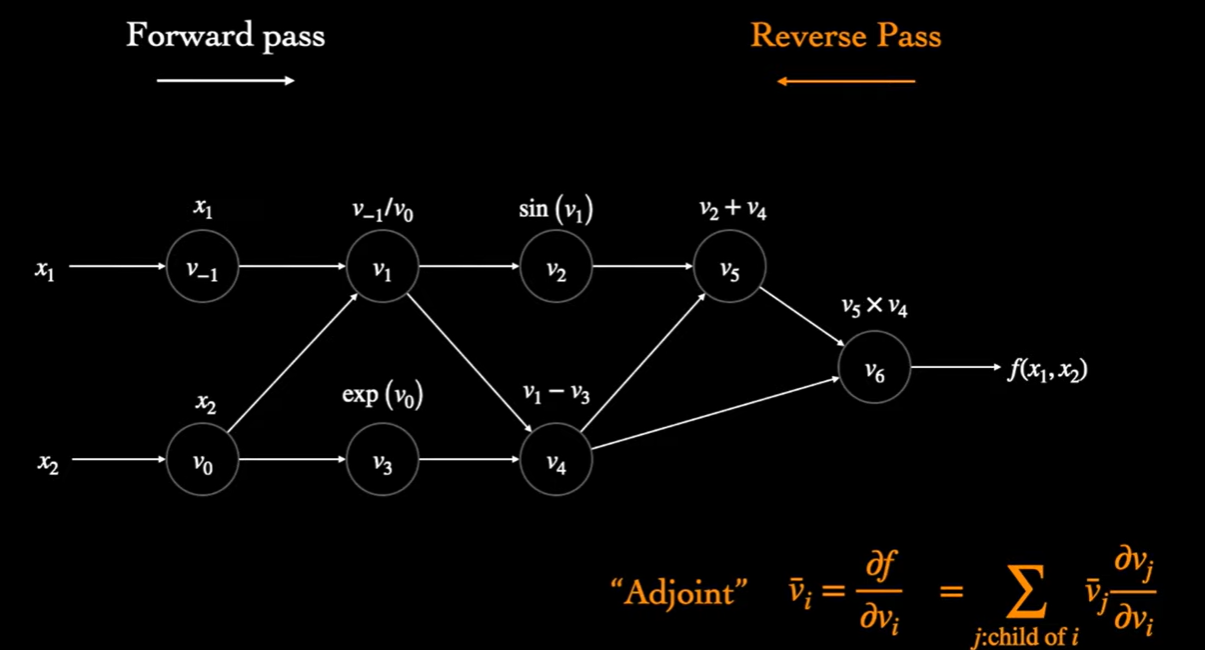
Đầu tiên sau đó  ∂�2/∂�1 vaà c  và cuối cùng

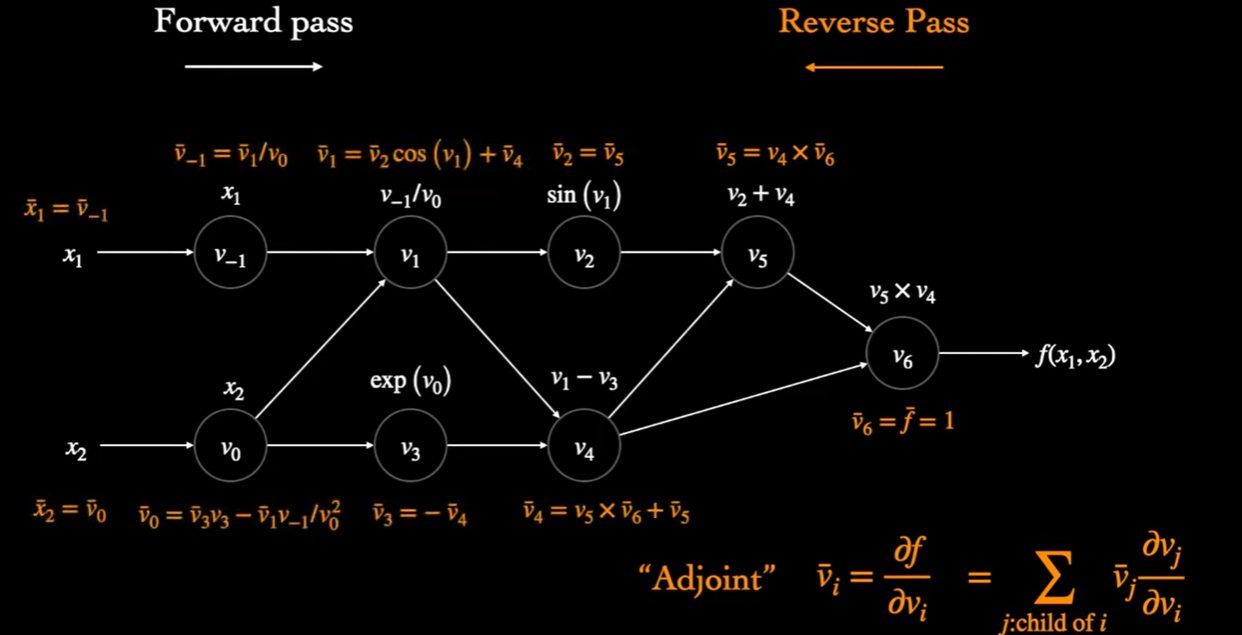
Biến phụ thuộc cần phải được đạo hàm được cố định và đạo hàm được tính đối với mỗi biểu thức con một cách đệ quy. Trong tính toán bằng bút và giấy, việc thay thế đạo hàm của các hàm bên ngoài trong quy tắc chuỗi được lặp đi lặp lại:

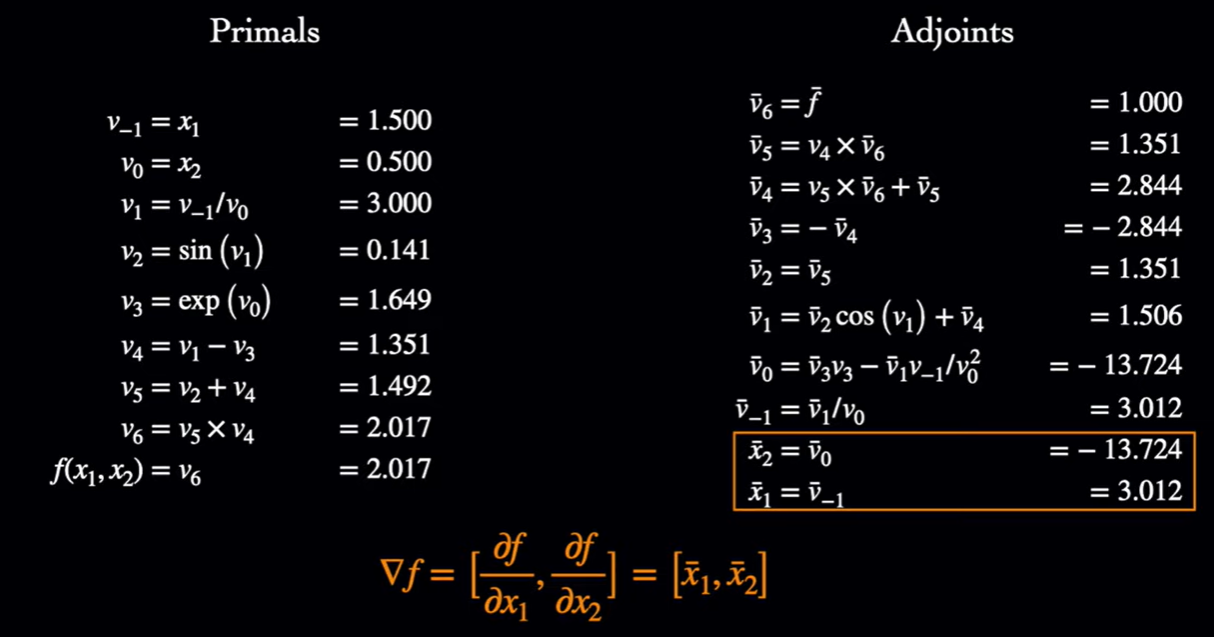


Reverse mode tính đệ quy:









Ưu điểm:

Phù hợp cho các mô hình có số lượng biến đầu vào lớn, vì nó chỉ yêu cầu một lần chạy qua mô hình để tính toán đạo hàm của toàn bộ hàm mất mát.

Hiệu suất tính toán tốt hơn so với chế độ tiến, đặc biệt là khi số lượng biến đầu ra lớn.

Nhược điểm:

Yêu cầu lưu trữ nhiều thông tin tạm thời trong quá trình tính toán, có thể tốn nhiều bộ nhớ.

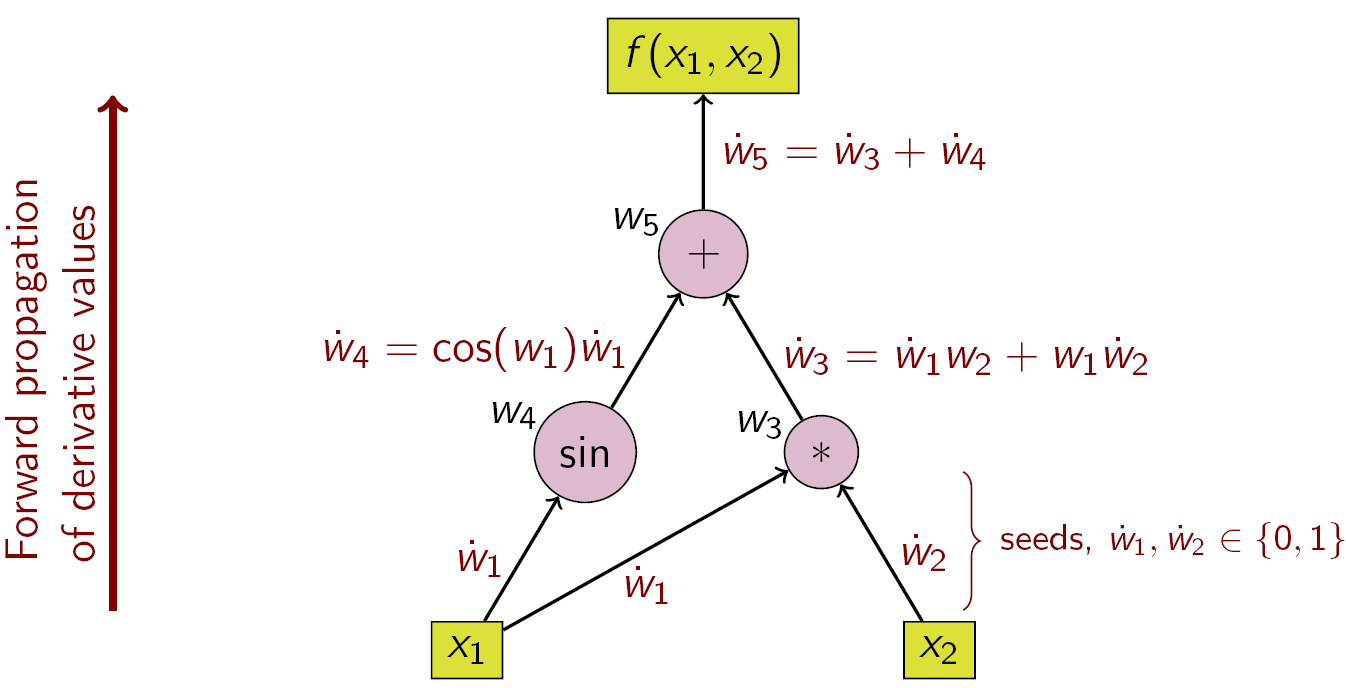
Cần phải chạy qua mô hình hai lần: một lần cho chạy tiến (forward) và một lần cho chạy lùi (backward).

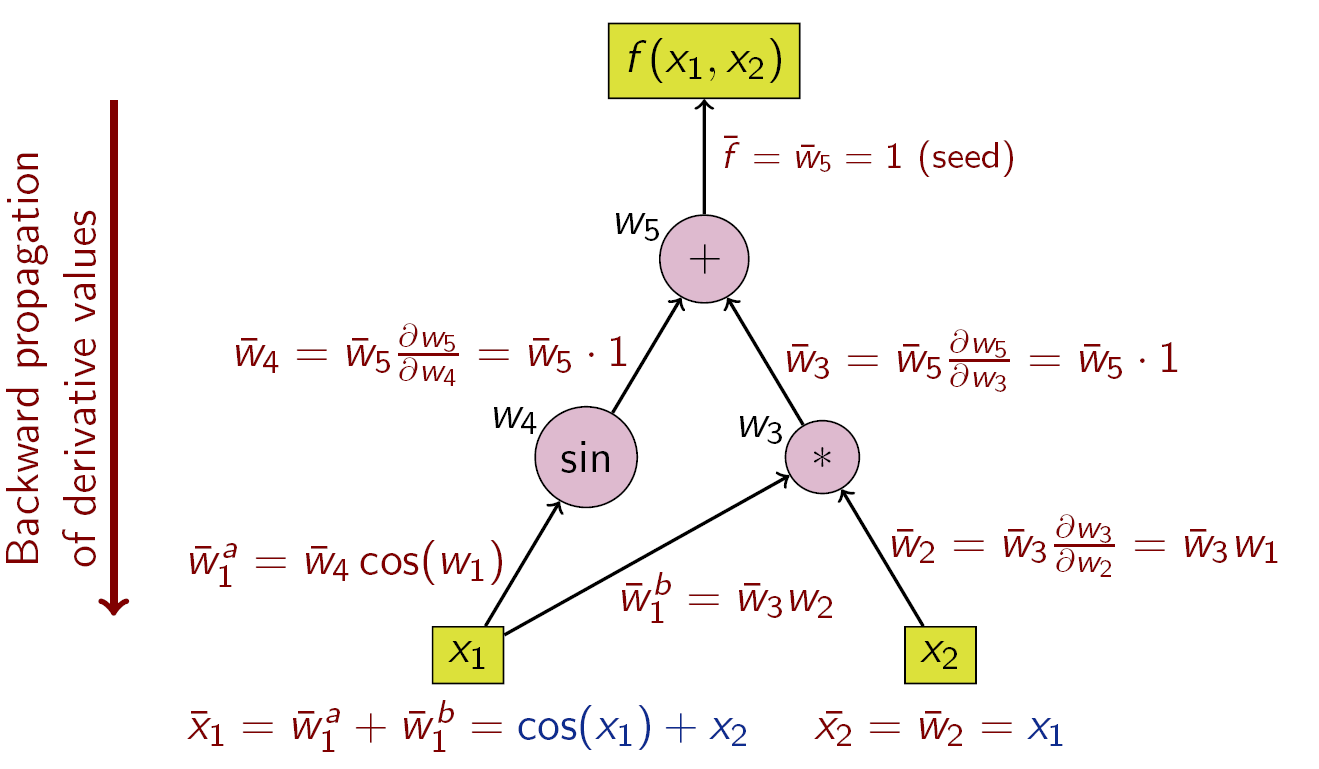
2.3 So sánh f : Rn → Rm

ops(f): operation count of f

|  |  |  |
| --- | --- | --- |
|  | Forward mode | Reverse mode |
| Space Complexity | O(1) | O(ops(f)) |
| Time Complexity (c<6) | n c ops(f) | m c ops(f) |
| n and m | n<<m | m<<n |
| Compute Hfv | fv (set x˙ = v) | f2v = Hfv |

1. Giải thích vai trò của computational graph trong AD và backpropagation:





Reverse mode AD và backpropagation có một lịch sử liên quan chặt chẽ.

Vai trò của computational graph trong AD:

Đồ thị tính toán (computational graph) đóng một vai trò quan trọng trong việc tính toán gradient của một hàm số. Dưới đây là một số vai trò chính của đồ thị trong AD:

+ Tổ chức dữ liệu và phép toán: Đồ thị tính toán cung cấp một cách cấu trúc để tổ chức dữ liệu và các phép toán trong quá trình tính toán hàm số. Các nút của đồ thị đại diện cho các biến hoặc các phép toán, trong khi các cạnh đại diện cho các phụ thuộc giữa chúng. Điều này giúp làm rõ quá trình tính toán và các liên kết giữa các biến và phép toán.

+ Tính toán đạo hàm theo chiều tiến (forward): Trong quá trình tiến, đồ thị tính toán được sử dụng để tính toán các giá trị của hàm số và các biến trung gian. Các giá trị này được tính toán và lưu trữ để sử dụng trong quá trình tính toán đạo hàm theo chiều lùi.

+ Tính toán đạo hàm theo chiều lùi (backward): Trong quá trình lùi lại, đồ thị tính toán được sử dụng để tính toán các đạo hàm của hàm số theo các biến đầu vào. Điều này được thực hiện bằng cách sử dụng luật chuỗi để lan truyền gradient từ đầu ra của hàm số đến các biến đầu vào thông qua các cạnh của đồ thị.

+ Hiệu quả tính toán gradient: Bằng cách sử dụng đồ thị tính toán, AD cho phép tính toán gradient của một hàm số với số lượng biến lớn một cách hiệu quả. Thay vì phải tính toán gradient bằng cách thủ công từng phần, AD tự động áp dụng luật chuỗi trên đồ thị tính toán để tính toán gradient một cách tự động và hiệu quả.

Backpropagation là một thuật toán cụ thể được sử dụng để tính toán các độ dốc một cách hiệu quả trong biểu đồ tính toán. Nó hoạt động bằng cách áp dụng đệ quy quy tắc chuỗi của phép toán vi phân từ đầu ra của biểu đồ đến các đầu vào của nó. Quá trình này bao gồm hai lần đi qua biểu đồ: một lần đi qua để tính toán đầu ra và một lần đi lại (backpropagation) để tính toán các độ dốc. Trong quá trình đi lại, các độ dốc được truyền ngược lại qua biểu đồ, và mỗi nút tính toán độ dốc của đầu ra của nó đối với các đầu vào của nó bằng cách sử dụng quy tắc chuỗi.

Vai trò của computational graph trong backpropagation:

Trong quá trình lan truyền ngược (backpropagation), một computational graph (đồ thị tính toán) có vai trò quan trọng trong việc tính toán đạo hàm của hàm mất mát theo các tham số của mạng nơ-ron.

+ Cụ thể, mạng nơ-ron có thể được biểu diễn dưới dạng một computational graph, trong đó các nút đại diện cho các phép toán và các cạnh đại diện cho các biến và giá trị tạm thời (intermediary values). Khi các phép toán được thực hiện, các giá trị tạm thời sẽ được tính toán và lưu trữ trong computational graph.

+ Khi quá trình lan truyền ngược bắt đầu, đạo hàm của hàm mất mát được tính bằng cách sử dụng nguyên lý chuỗi (chain rule) trên computational graph. Cụ thể, các đạo hàm riêng của các phép toán trong mạng nơ-ron được tính và lan truyền ngược từ các lớp cuối cùng đến các lớp đầu tiên. Quá trình này cho phép tính toán đạo hàm của hàm mất mát theo các tham số của mạng, giúp cập nhật các trọng số và các siêu tham số (hyperparameters) thông qua các thuật toán tối ưu hóa như gradient descent.

+ Do đó, computational graph có vai trò quan trọng trong quá trình lan truyền ngược bằng cách cung cấp một cấu trúc tổ chức để tính toán và lan truyền đạo hàm một cách hiệu quả.

1. Practical Implementation:

* Choose a deep learning framework (PyTorch) and implement a simple neural network model from scratch using Autograd capabilities.
* Define a custom loss function and explore how Autograd computes gradients for this function.
* Experiment with different optimization algorithms (e.g., SGD, Adam) and observe their effects on training convergence and performance.

1. Experimental Analysis:

* Conduct experiments to analyze the efficiency and numerical stability of Autograd-based gradient computations.
* Investigate the impact of batch size, learning rate, and network architecture on gradient computation and training dynamics.

**Question 2 (5 points):**

Implement CNN-LSTM models for the problem of Image Captioning with at least two options: with and without using attention mechanism for the decoding process. Training the models on Colab environments using GPU and CPU only.