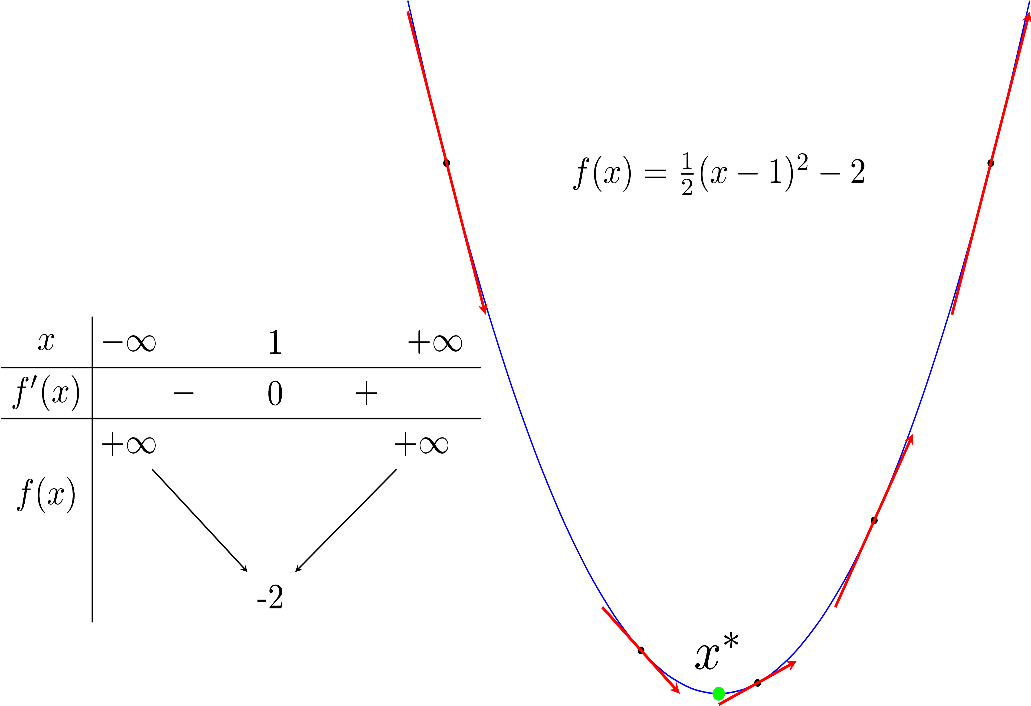
**Báo cáo tiến trình học tập tuần 3**

1. **Gradient Descent**
2. **Giới thiệu bài toán**

* Do việc đạo hàm với các phương trình phức tạp và có số chiều lớn, hoặc có quá nhiều điểm dữ liệu. Hướng tiếp cận phổ biển là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm bài toán, sau đó dùng phép toán lặp để tiến dần về điểm cần tìm.

1. **Gradient Descent cho hàm 1 biến**



Giả sử là điểm ta tìm được sau vòng lặp thứ t. Ta cần tìm một thuật toán để đưa về càng gần càng tốt.

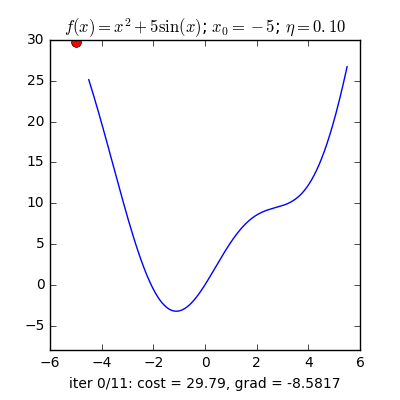
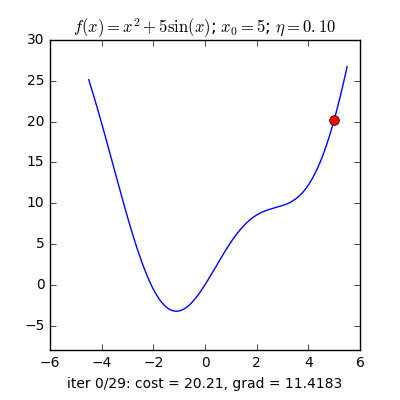
* Nếu đạo hàm của hàm số tại : thì nằm về phía phải so với . Để điểm tiếp theo gần hơn với , ta cần di chuyển về phía trái. Nói cách khác, chúng ta cần di chuyển ngược dấu với đạo hàm:

Trong đó là một đại lượng ngược dấu với đạo hàm.

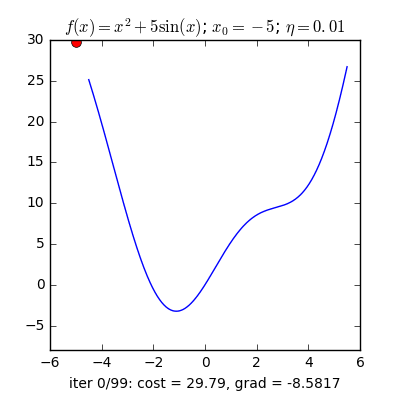
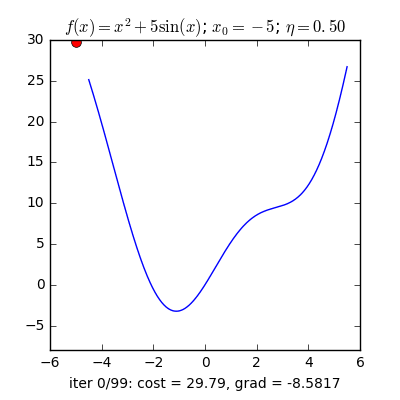
* càng xa về phía bên phải thì càng lớn hơn 0. Vậy lượng di chuyển tỉ lệ thuận với .
* Từ đó, ta có một công thức cập nhật là:

1. **Một số lưu ý khi sử dụng Gradient Descent**
2. Điểm khởi tạo

* Với hai điểm khởi tạo khác nhau là 5 và -5 thì tốc độ hội tụ của mô hình là khác nhau.

1. Learning rate khác nhau

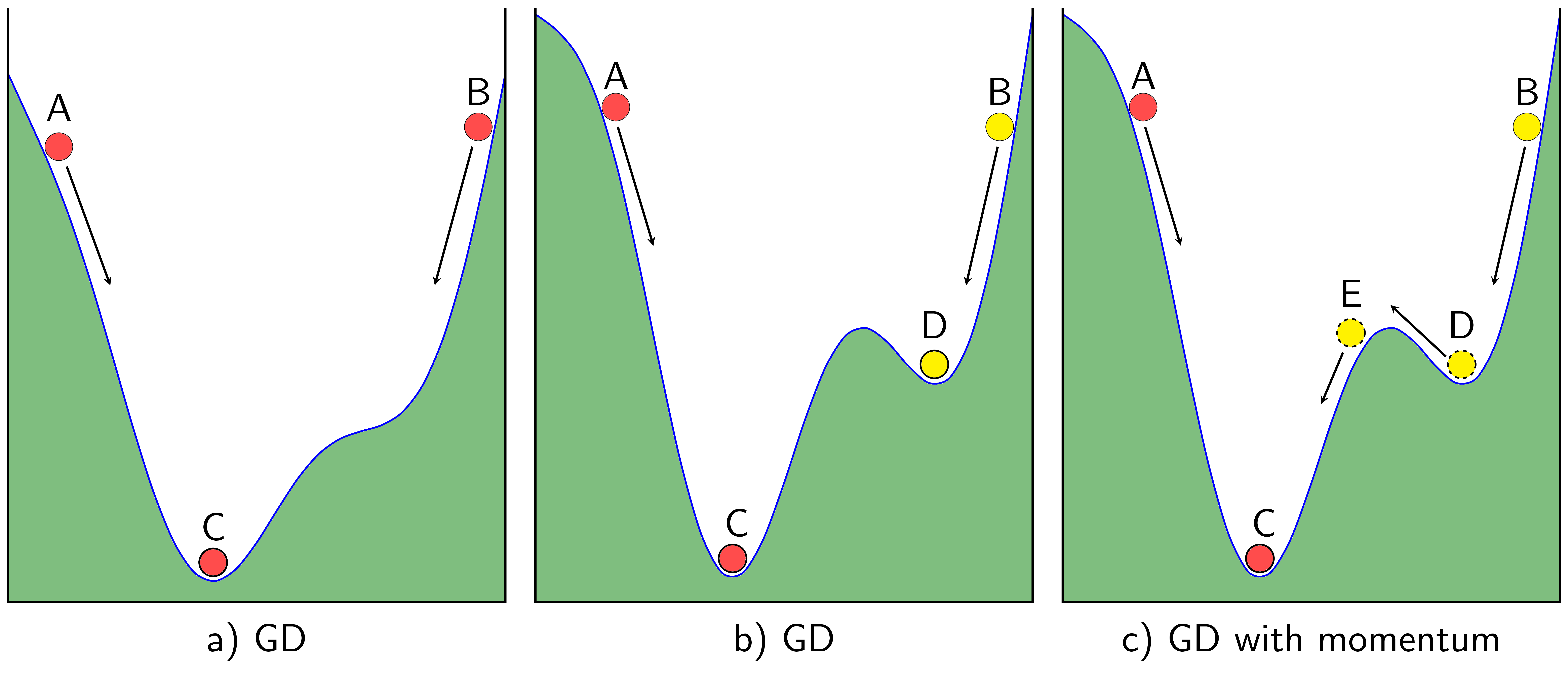
 

* Với , tốc độ hội tụ rất chậm. Trong thực tế, khi việc tính toán trở nên phức tạp, *learning rate* quá thấp sẽ ảnh hưởng tới tốc độ của thuật toán rất nhiều, thậm chí không bao giờ tới được đích.
* Với , thuật toán tiến sát nhanh tới đích sau vài vòng lặp. Tuy nhiên, thuật toán không hội tụ được vì *bước nhảy* quá lớn, khiến nó cứ *quẩn quanh* ở đích.
* Tùy vào một số bài toán, GD có thể làm việc hiệu quả hơn bằng cách chọn ra *learning rate* phù hợp hoặc chọn *learning rate* khác nhau ở mỗi vòng lặp.

1. **Gradient Descent cho hàm nhiều biến**

Tương tự như hàm 1 biến , ta có công thức cập nhật là:

1. **Các thuật toán tối ưu Gradient Descent**
2. Momentum



* Ý tưởng là làm thế nào cho hòn bi đi xuống được C mà không dừng lại ở D, như trong vật lý, nếu vận tốc đủ lớn, thì khi lăn đến D, nó đang có đà và vượt qua rãnh D để tiếp tục đến C (đây là điều chúng ta mong muốn) => Momentum dựa trên nguyên lí này.
* Nếu chúng ta coi đại lượng thay đổi ở thời điểm t như là vận tốc trong vật lý, thì vị trí mới của hòn bi sẽ là :.
* vừa mang thông tin độ dốc, vừa mang thông tin của đà, tức vận tốc trước đó (chúng ta coi như vận tốc ban đầu = 0, nên ta có:
* Trong đó thường được chọn là một khoảng 0.9, vị trí mới của hòn bi được xác định như sau:

1. Nesterov accelerated gradient (NAG)

* Đây là một cải tiến của momentum, vì momentum có một hạn chế là khi gần tới đich nó vẫn mất khá nhiều thời gian trước khi dừng lại.
* Ý tưởng chính là nhìn trước một bước. Nghĩa là nó sẽ trừ đi một lượng đà trước đó, rồi xem nên di chuyển như thế nào ở bước hiện tại, chứ không như momentum, di chuyển xong rồi mới chỉnh. Nhờ đó, phần đạo hàm sẽ di chuyển chính xác hơn ở bước tiếp theo.
* Công thức cập nhật của NAG là:
* Sau đó, cập nhật :

1. **Các thuật toán khác**
2. Batch Gradient Descent

* Thuật toán Gradient Descent còn được gọi là Batch Gradient Descent. Batch ở đây được hiểu là tất cả, tức khi cập nhật , chúng ta sử dụng tất cả các điểm dữ liệu .

1. Stochastic Gradient Descent

* Khác với GD, SGD tính đạo hàm tại một điểm dữ liệu bất kì rồi cập nhật cho toàn bộ
* Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là một epoch.
* Với GD thông thường thì mỗi epoch ứng với 1 lần cập nhật , còn với SGD thì mỗi epoch ứng với N lần cập nhật .
* SGD phù hợp với các bài toán có lượng CSDL lớn, hoặc các bài toán có mô hình thay đổi liên tục, tức online learning.

**Thứ tự lựa chọn điểm dữ liệu**

* Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD.
* Một cách toán học, quy tắc cập nhật của SGD là:
* Hiểu SGD đơn giản là thay vì thận trọng mỗi lần cập nhật như GD (duyệt tất cả rồi mới cập nhật), thì SGD cập nhật liên tục (làm nhiều quen tay), số lượng cập nhật của SGD rất lớn nhưng epoch thì rất ít.

1. Mini-batch Gradient Descent

* Khác với SGD, Mini-batch sử dụng một số lượng n lớn hơn 1 (vẫn nhỏ hơn tổng số dữ liệu N rất nhiều).
* Mỗi mini-batch có n điểm dữ liệu (mini-batch cuối có thể ít điểm dữ liệu hơn nếu N không chia hết cho n), công thức có thể viết dưới dạng:
* Các thuật toán tối ưu như Momentum, NEG, có thể áp dụng vào đây được.
* Mini-batch GD được sử dụng trong hầu hết các thuật toán Machine Learning, đặc biệt là trong Deep Learning. Giá trị n thường được chọn là khoảng từ 50 đến 100.

1. **Stopping Criteria (điều kiện dừng)**

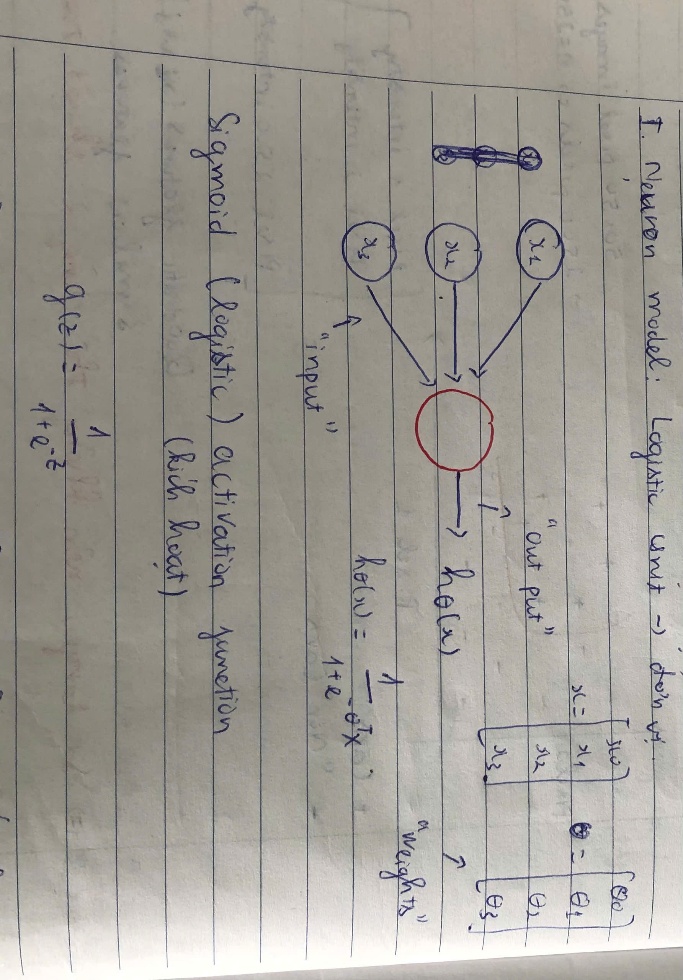
Trong thực nghiệm, có một vài phương pháp như dưới đây:

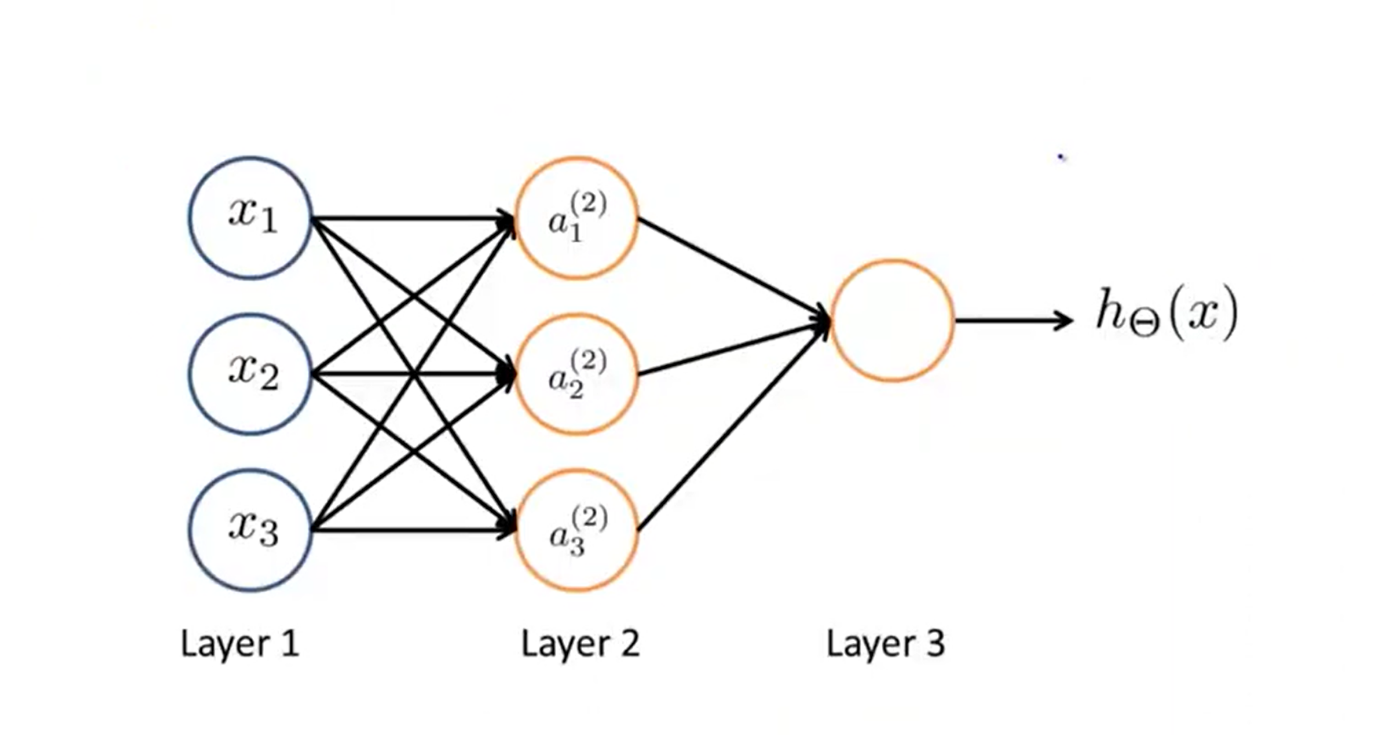
* Giới hạn số vòng lặp: đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
* So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Phương pháp này cũng có một nhược điểm lớn là việc tính đạo hàm đôi khi trở nên quá phức tạp (ví dụ như khi có quá nhiều dữ liệu), nếu áp dụng phương pháp này thì coi như ta không được lợi khi sử dụng SGD và mini-batch GD. (*Vì các cách này nó trả về các điểm quanh điểm hội tụ nên việc kiểm tra gradient là không đáng tin*)
* So sánh giá trị của hàm mất mát của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Nhược điểm của phương pháp này là nếu tại một thời điểm, đồ thị hàm số có dạng *bẳng phẳng* tại một khu vực nhưng khu vực đó không chứa điểm local minimum (khu vực này thường được gọi là saddle points), thuật toán cũng dừng lại trước khi đạt giá trị mong muốn.
* Trong SGD và mini-batch GD, cách thường dùng là so sánh nghiệm sau một vài lần cập nhật.

1. **Neural Network**
2. **Giới thiệu**

* Neural Network, hay mạng nơ-ron nhân tạo, là một mô hình học máy được lấy cảm hứng từ cấu trúc não bộ. Khác với các mô hình tuyến tính truyền thống, Neural Network có khả năng học và biểu diễn các quan hệ phi tuyến phức tạp.

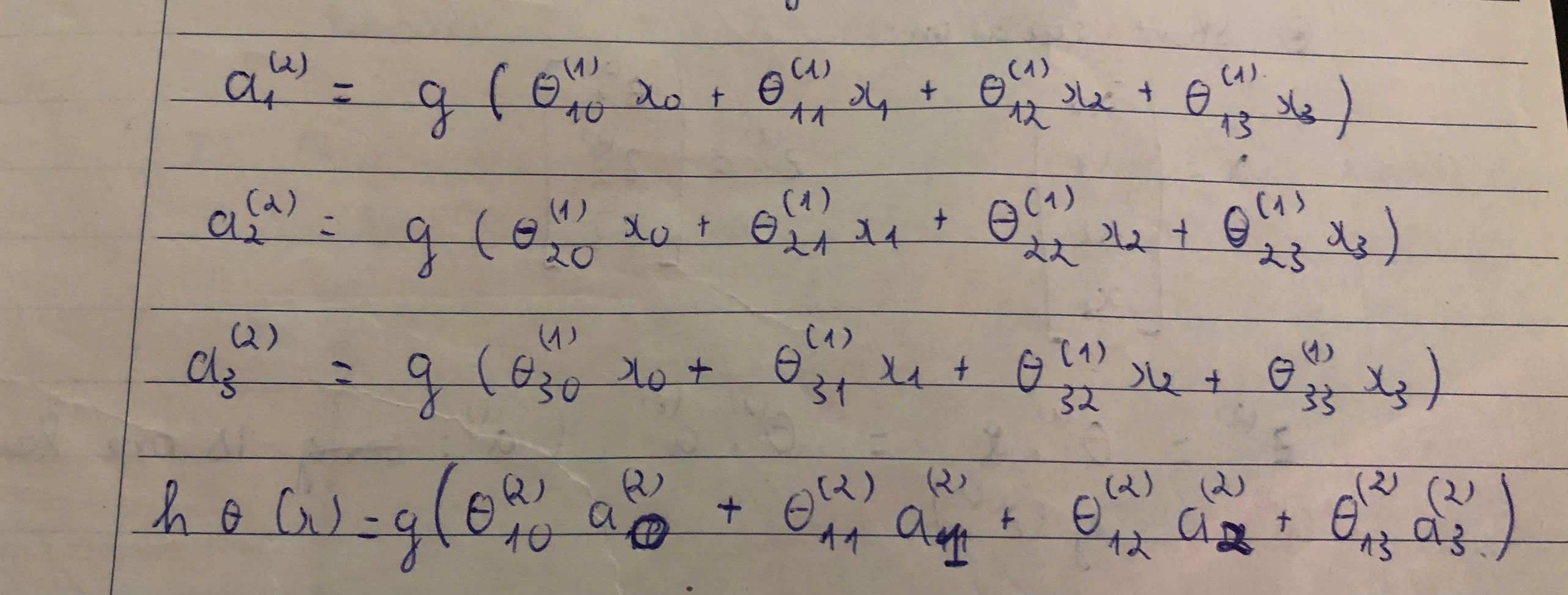
1. **Neural model: Logistic unit**



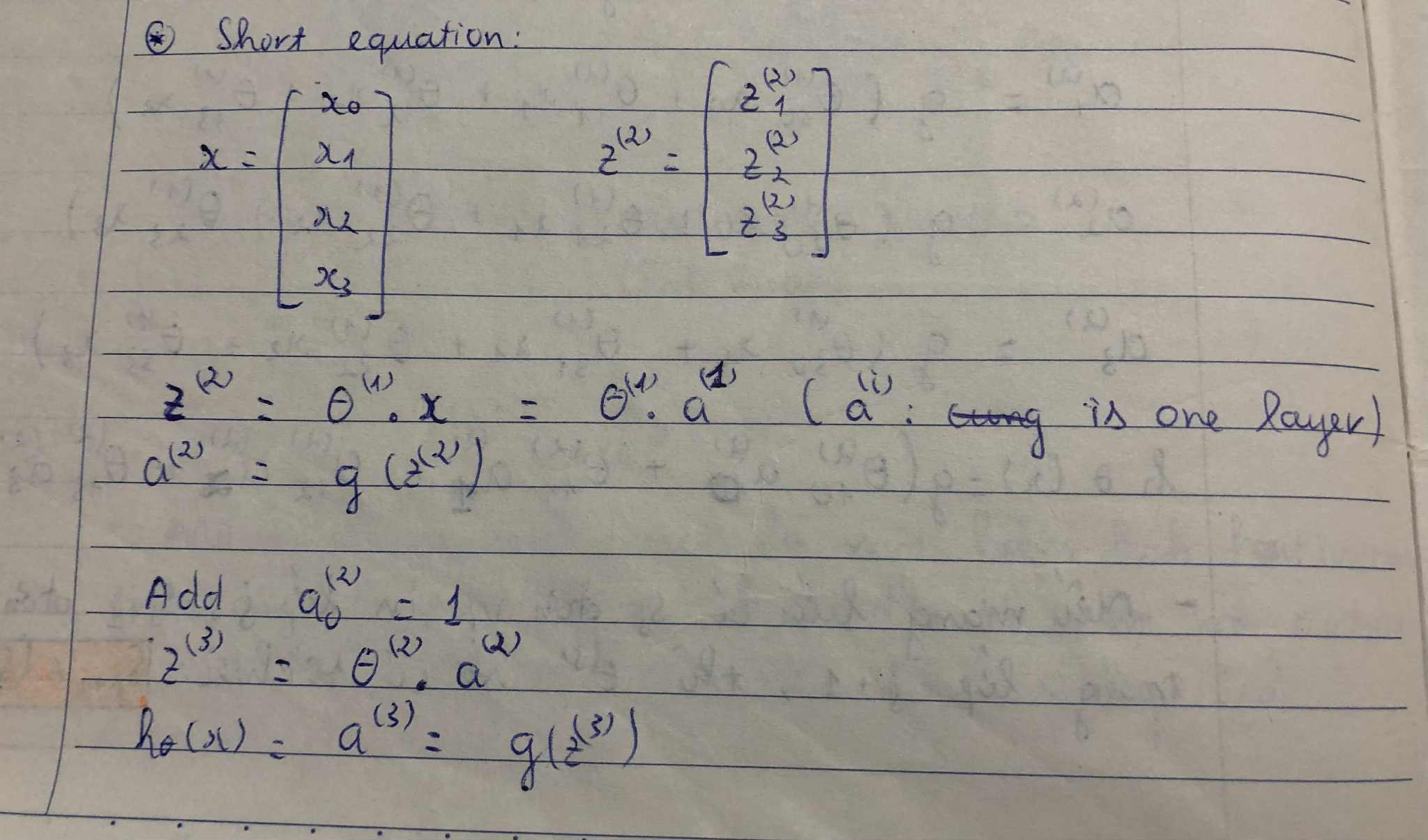
1. **Neural Network và Forward propagation**Lưu ý:

* Neural Network gồm 3 phần là input layer, output layer và hidden layer.
* được gọi là “activation” của đơn vị i trong lớp j.
* Các activation này tùy thuộc vào người build lựa chọn.
* Để tránh mô hình học cùng một điều vì tính đối xứng, chúng ta khởi tạo w là random để phá vỡ tính đối xứng của nó.

Forward propagation:

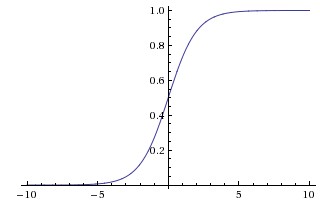
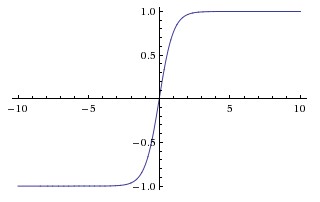
****

Cách biểu diễn vector:



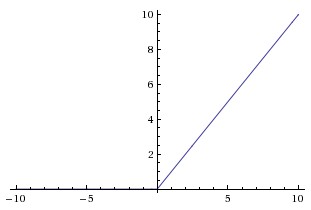
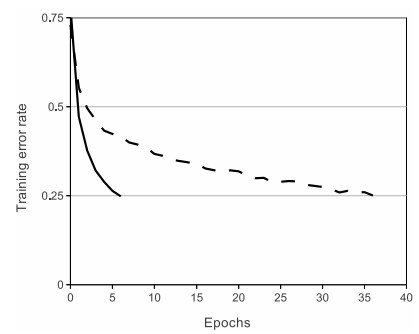
Một số activation thường được sử dụng:

1. Sigmoid và tanh

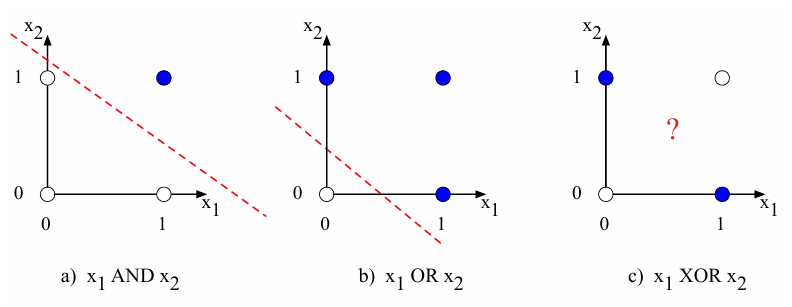
* Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật.

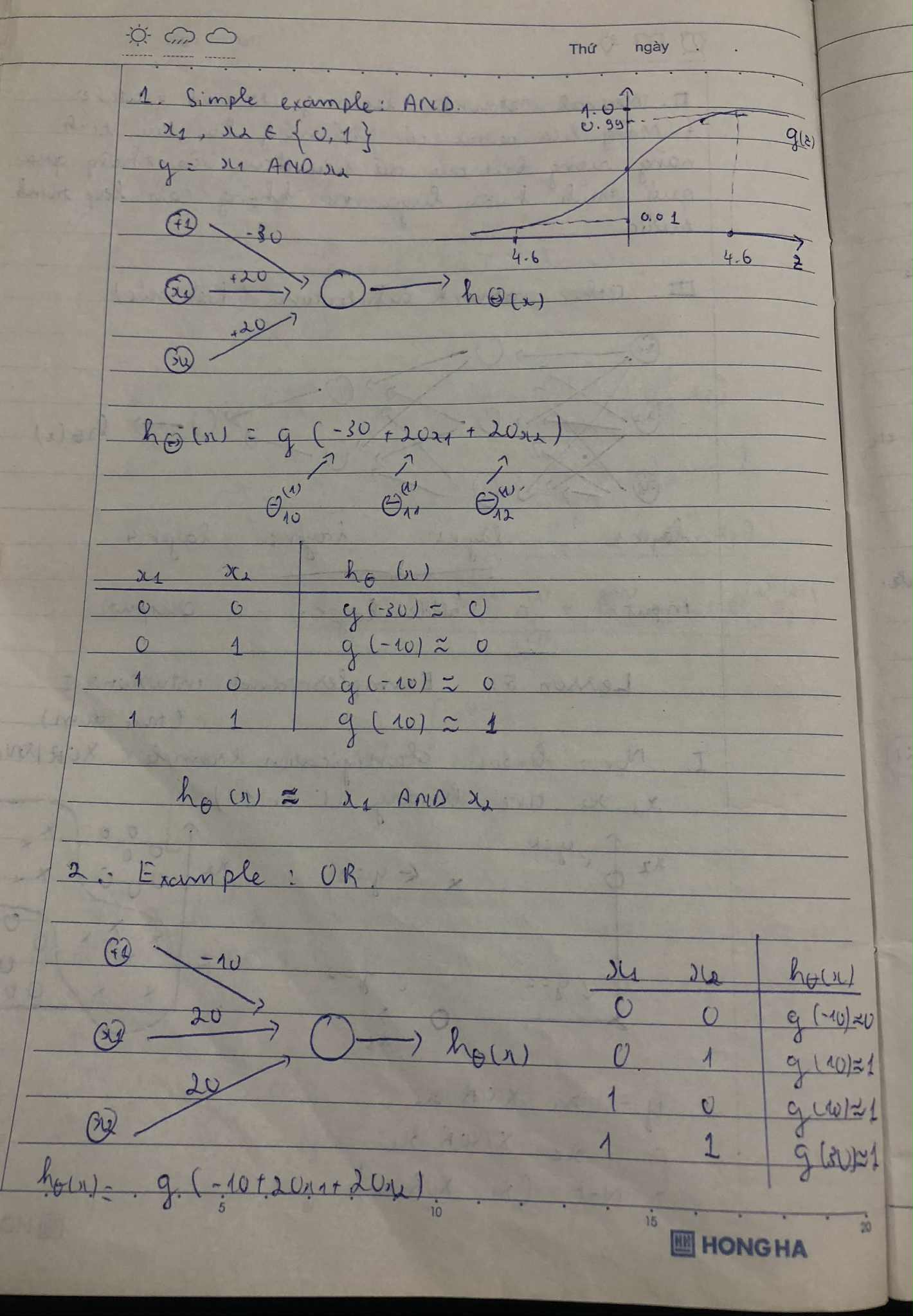
1. ReLu (Rectified Linear Unit)

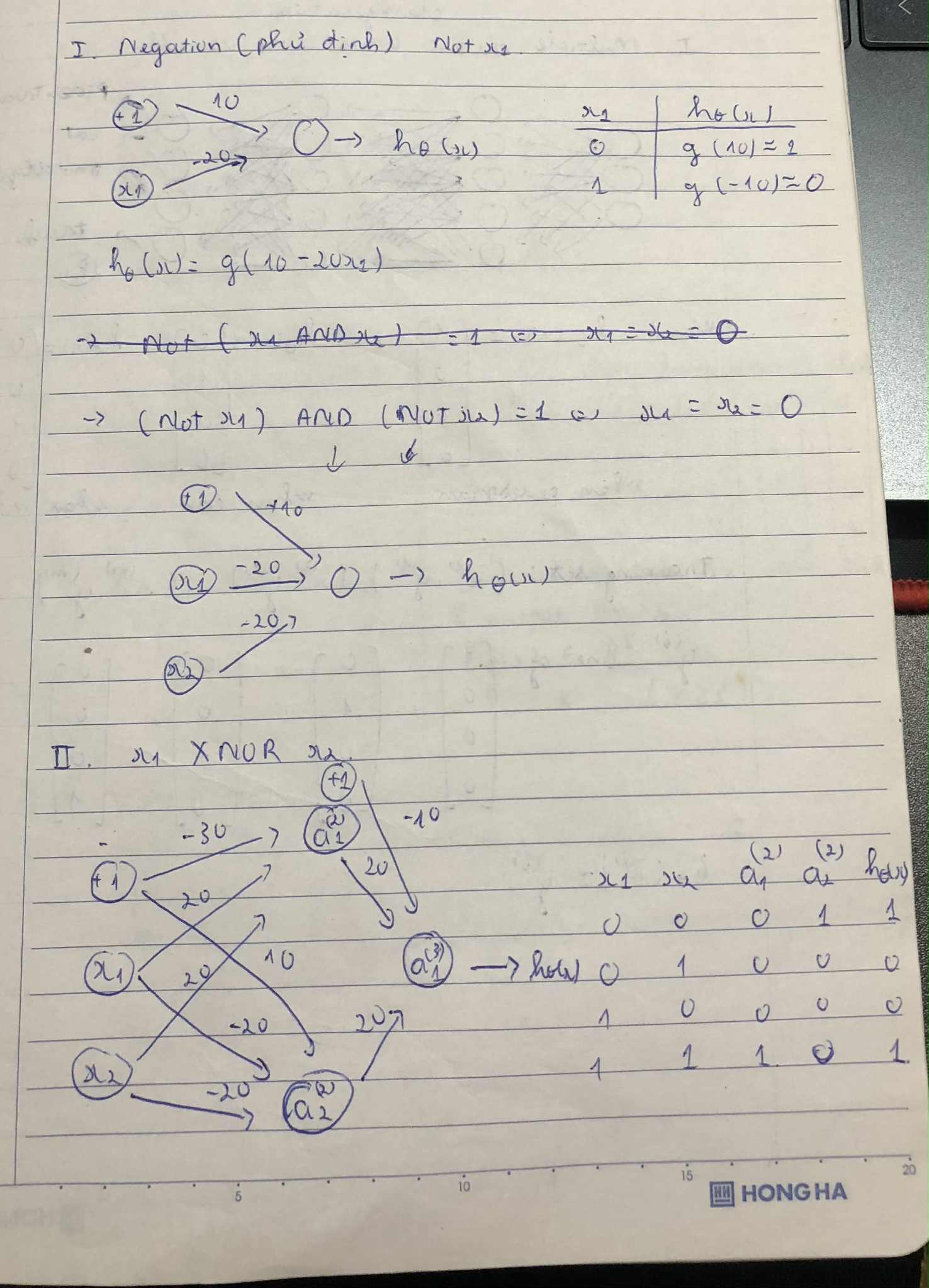
 

* Nó có công thức toán học là: **.**
* Relu được chứng minh giúp cho việc training các Deep Networks nhanh hơn rất nhiều.
* HÌnh bên phải là so sanh sự hội tụ của SGD khi sử dụng 2 actitvation là ReLu và tanh.
* Sự tăng tốc này được cho là vì ReLu được tính toán gần như tức thời và gradient của nó cũng được tính cực nhanh với gradient bằng 1 nếu đầu vào lớn hơn 0, bằng 0 nếu đầu vào nhỏ hơn 0.

1. **Trực quan Neural Network bằng bài toán XNOR**

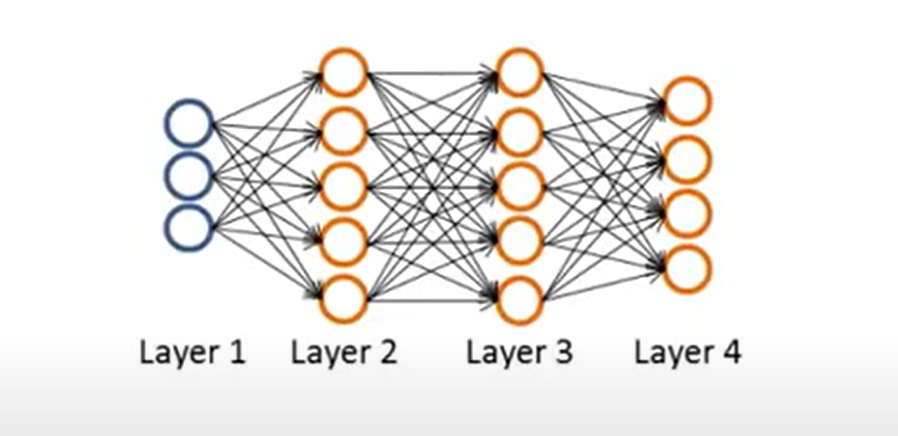
****





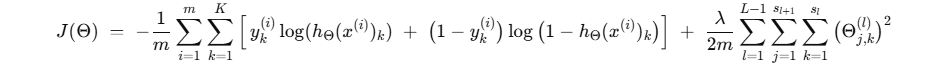
1. **Backpropagation**
2. Giới thiệu mô hình

* Giả sử ta có mạng neural như sau, activation là hàm sigmoid:

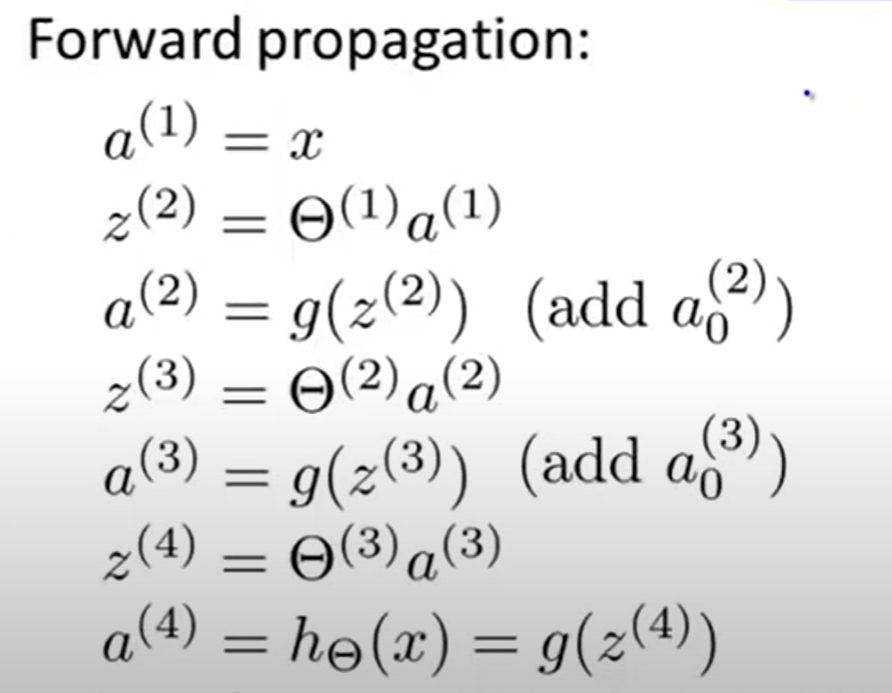
****

1. Hàm cost function

* Ta có hàm cost function với activation là hàm sigmoid:



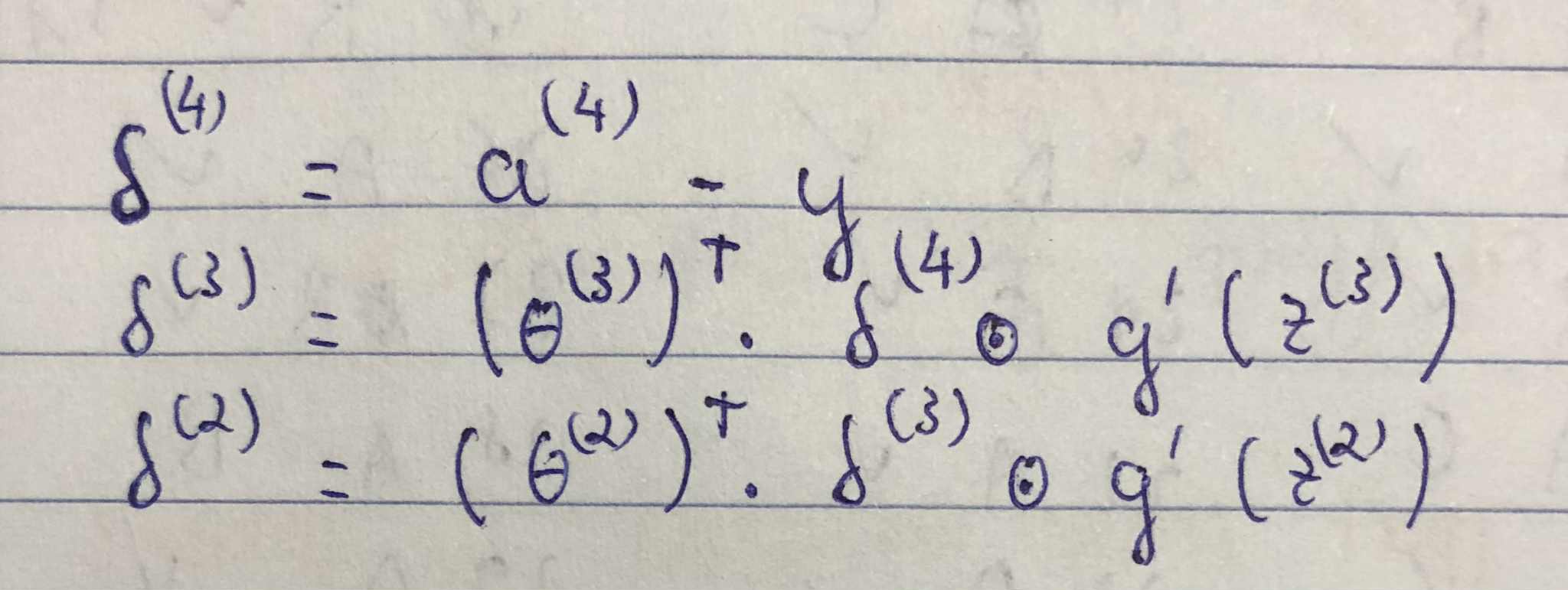
1. Forwardpropagation



1. Công thức cập nhật

* được định nghĩa là “sai số” tại lớp i, cụ thể là:

Trong đó L là hàm loss function.



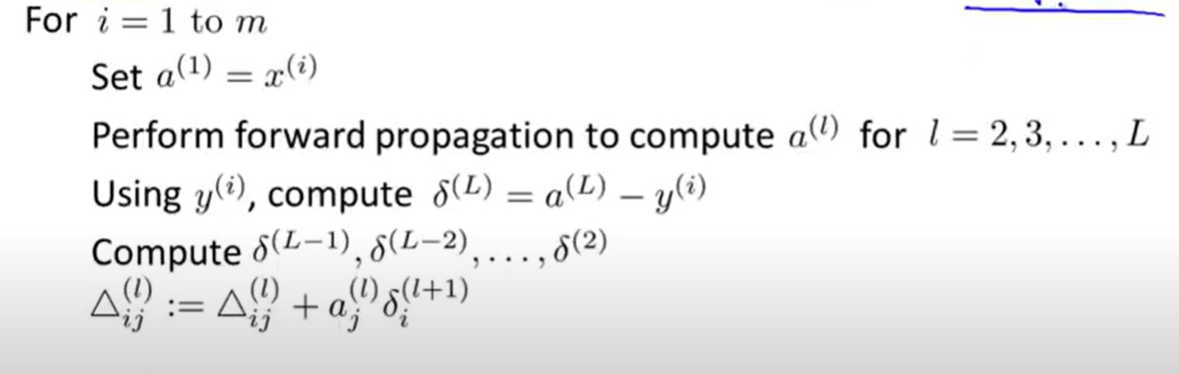
* Giả sử ta có: thuật toán backpropagation sẽ thực hiện việc tính ngược các chênh lệnh này, dựa trên chênh lệnh của lớp trước bằng công thức tính sai số kết hợp với chain rule.
* Trong đó ⊙ tức lấy từng thành phần của hai vector nhân với nhau để được vector kết quả.

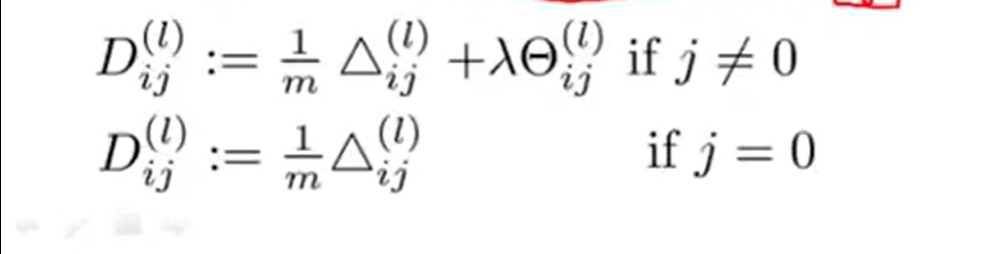
**Cách thực hiện tính delta3:**

* Ta có:
* Trong đó
* Do đó:
* Vì có activation là hàm sigmoid, nên
* Không có error ở lớp 1 vì nó là input.
* Đạo hàm hàm mất mát J theo trọng số:

1. Cập nhật

Khởi tạo (cho tất cả l, i, j)

****

****

* **Công thức cập nhật**

