# ADALINE Network

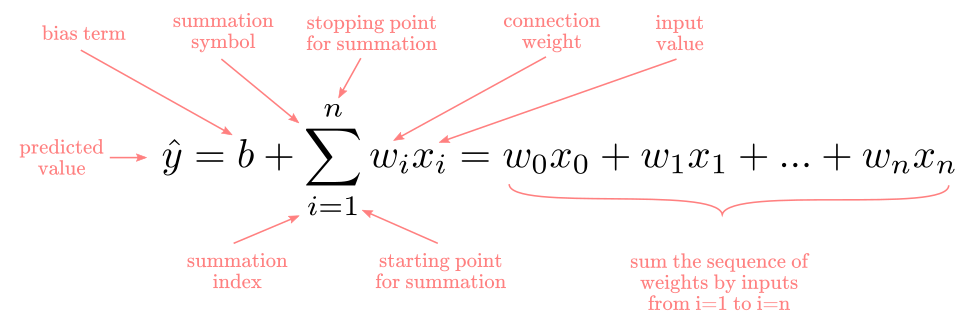
## Hình thức hóa toán học

Một cách toán học, ADALINE được mô tả như sau:

* Một hàm tuyến tính tổng hợp các tín hiệu đầu vào.
* Một thủ tục học để điều chỉnh trọng số kết nối.

## Hàm tổng hợp tuyến tính

Hàm tổng hợp tuyến tính tương tự như phương pháp perceptron.



Hàm quyết định ngưỡng

Với là đầu ra của hàm tuyến tính.

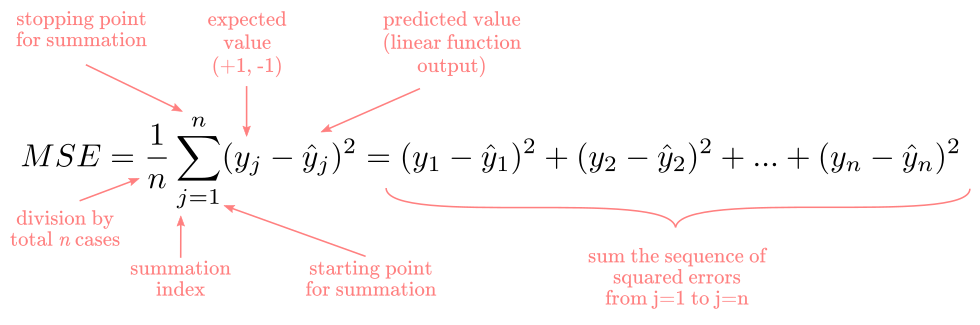
## Khác biệt cơ bản của Perceptron và ADALINE

Khác biệt là **thủ tục học tập để cập nhật trọng số** của mạng. Perceptron cập nhật các trọng số bằng cách tính toán hiệu của các lớp giá trị dự kiến (excepted) và dự đoán (predicted). Nói cách khác, preception luôn so sánh +1 và -1 (predicted values) thành +1 hay -1 (expected values). Một hệ quả quan trọng của perceptron là nó *chỉ học khi xuất hiện lỗi*. Ngược lại, ADALINE tính toán sự khác biệt giữa lớp giá trị (+1 hay -1) và giá trị đầu ra liên tục từ hàm tuyến tính, cái mà có thể là bất kì số thực nào. Điều này rất quan trọng vì có nghĩa là ADLINE có thể *học ngay cả khi không có lỗi phân loại nào được thực hiện*. Đây là hệ quả thực tế rằng các lớp dự đoán không ảnh hưởng đến việc tính toán lỗi. Vì ADALINE luôn học và perceptron chỉ học sau khi có lỗi, ADALINE sẽ tìm ra giải pháp nhanh hơn perceptron cho cùng một vấn đề.

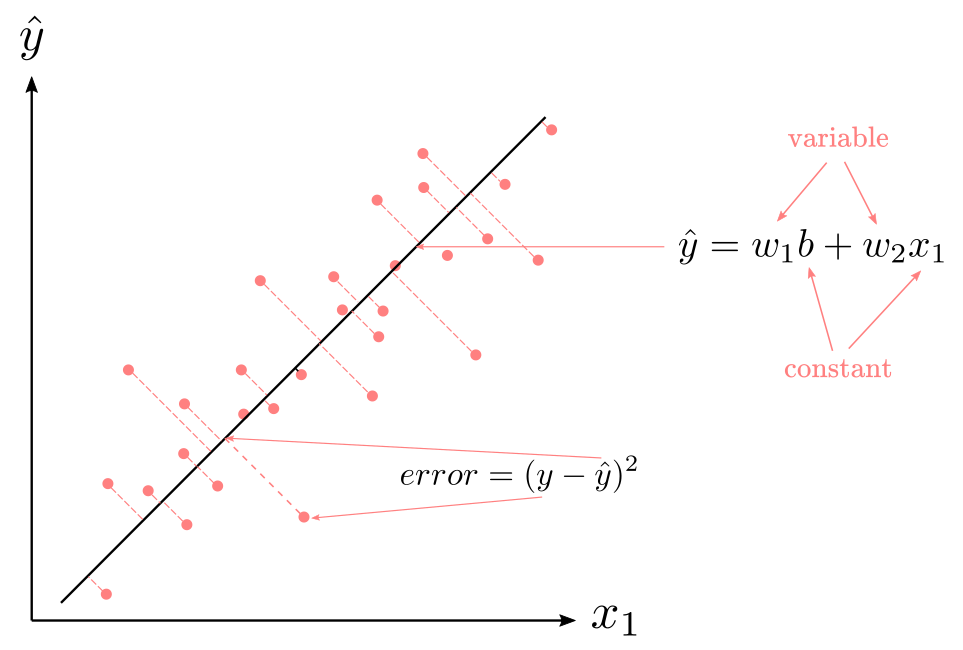
## Mặt lỗi của ADALINE

Trước khi tiến đến định nghĩa của mô hình thủ tục học của ADALINE, ta sẽ khám phá ngắn gọn của ý nghĩa của “giảm thiểu giá trị trung bình của tổng các lỗi bình phương”. Tương tự như least-squares-method trong phân tích hồi quy.

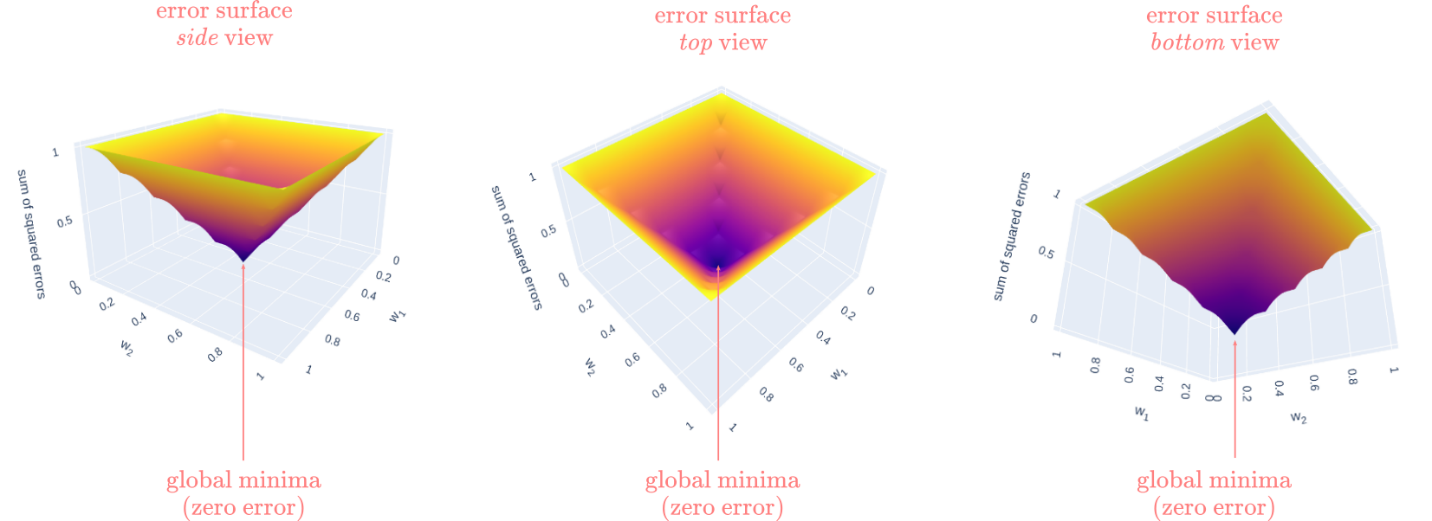
Trong một lần lặp, lỗi trong ADALINE được tính toán như sau , nói cách khác, nó là bình phương hiệu số của expected value và predicted value. Quá trình so sánh các giá trị của expected và predicted được lặp lại trong tất cả các trường hợp, đến , trong một tập dữ liệu nhất định. Khi chúng ta thêm sự khác biệt bình phương cho toàn bộ tập dữ liệu và chia cho tổng số, chúng ta gọi là mean of squared errors (MSE):



Hình sau cho thấy ví dụ trực quan với một yếu tố dự đoán. Trục hoành thể hiện predictor (hay feature), trung tung biểu thị giá trị dự đoán (predicted value) , và các chấm màu hồng đại diện cho các giá trị dự kiến (real data points). Mặt phẳng Đề-cát (Cartes plane) là . Điều quan trọng là giao điểm *(intercept*) (tức là nơi đường bắt đầu) và độ dốc (*slope*) (hay là độ nghiêng) của đường được xác định bởi các trọng số và . Các giá trị và được cho trước, *không thay đổi,* vì vậy không ảnh hưởng đến hình dạng của đường.



Mục tiêu của thuật toán bình phương nhỏ nhất (least square algo.) là tạo ra càng ít lỗi càng tốt. Điểu này tương đương với việc tìm đường thẳng phù hợp nhất với các điểm trọng mặt phẳng Đề-cát. Vì các trọng số là *giá trị duy nhất* chúng ta có thể điều chỉnh để thay đổi hình dạng của đường, nên **các trọng số khác nhau sẽ tạo ra các sai số bình phương khác nhau**. Chúng ta cần tìm cực tiểu trong một bề mặt lỗi. Tưởng tượng rằng chúng ta đang cố gắn tìm tập hợp các trọng số, và sẽ tạo ra sai số bình phương nhỏ nhất. Trọng số của chúng ta có thể nhận các giá trị nằm trong khoảng từ 0 đến 1 (hay 0% đến 100% theo tỉ lệ). Bây giờ chúng ta sẽ quyết định vẽ các giá trị lỗi bình phương với tất cả các kết hợp có thể có của và . Hình sau cho thấy bề mặt kết quả:



Chúng ta gọi nó là bề mặt lỗi (*error surface*). Trong trường hợp này, hình dạng của bề mặt lỗi tương tự như hình nón hay kim tự tháp đặt biệt là một điểm tại đó hoàn toàn giảm xuống 0 ở dưới cùng của đối tượng. Trong toán học, điểm đó được gọi là cực tiểu toàn cục (**global minima**). Trong trường hợp này, khi một tập các trọng số xác định một điểm duy nhất có lỗi bằng 0, thì gọi là bài toán tối ưu lồi (convex optimization problem).

**Tất cả các mạng thần kinh có thể coi là việc giải quyết vấn đề tối ưu hóa**,thông thường, trong không gian đa chiều, với hàng trăm hay hàng triệu trọng số được điều chỉnh để tìm ra giải pháp tốt nhất.

Tin xấu là hầu hết các vấn đề đáng để giải quyết trong khoa học nhận thức đều không lồi (nonconvex), có nghĩa rằng việc tìm cái gọi là cực tiểu toàn cụctrở nên vô cùng khó khăn, và hầu hết trong các trường hợp đều không được đảm bảo. Trong trường hợp không gian ba chiều, thay vì có bề mặt nổi như một hình nón đẹp mắt, ta thu được một cái gì đó giống như cảnh quan phức tạp của núi và thung lũng.

A picture containing colorfulness, magenta

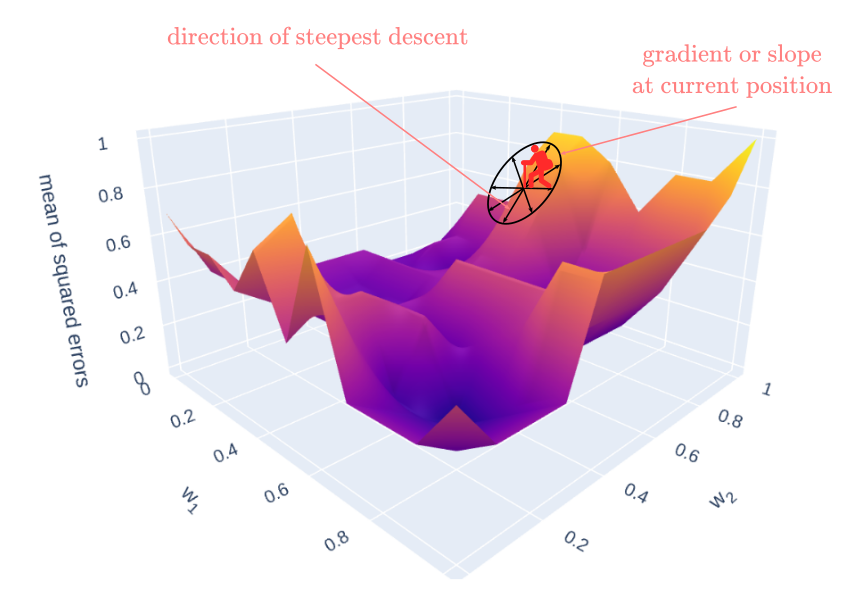
Description automatically generated

Bây giờ thay vì có được một diểm duy nhất là cực tiểu toán cục mà ở đó lỗi ở mức tối thiểu, ta thu được *các điểm thấp* hay *“thung lũng”* ở các phần (section) khác nhau trên bề mặt. Những “thung lũng” đó được gọi là cực tiểu cục bộ (**local minima**), hay điểm sai số tối thiểu cho phần đó. Lý tưởng nhất thì ta sẽ tìm cực tiểu toàn cục, tuy nhiên với một cảnh quan như vậy việc tìm kiếm có thể rất khó khăn và chậm chạp.

## Thủ tục học

Bây giờ, ta biết rằng ta muốn tìm tập các tham số để giảm thiểu hóa MSE. ADALINE tiếp cận điều này bằng cách sử dụng cái gọi là thuật toán giảm độ dốc (**gradient descent algorithm**). Đối với các bài toán lồi (bề mặt như hình nón), việc giảm độ dốc để đảm bảo tìm ra cực tiểu toàn cục. Đối với các bài toán không lồi, việc giảm độ dốc chỉ để tìm ra cực tiểu cục bộ, có thể hoặc không là cực tiểu toàn cục. Tại thời điểm này, ta chỉ thảo luận về các vấn đề tối ưu lồi.

Tưởng tượng rằng bạn đang đi bộ trên đỉnh một ngọn núi ở sườn của thung lũng. Mục tiêu của chúng ta là xuống đáy của thung lũng. Theo logic, bạn sẽ muốn đi bộ xuống dốc qua sườn đồi cho đến khi đến đáy. Trong bối cảnh đào tạo mạng nơ-ron, nó gọi là “giảm dốc” (descending a gradient). Bây giờ, thật tuyệt nếu ta muốn làm điều này *một cách hiệu quả*, nghĩa là đi theo con đường sẽ đưa ta đến thung lũng nhanh hơn. Điều này tương dương với việc dọc theo bề mặt lỗi mà hướng độ dốc (độ nghiêng là lớn nhất). Trong bối cảnh tối ưu hóa, chúng ta có thể sử dụng quy tắc chuỗi của phép tính (chain-rule of calculus) để ước tính độ dốc và điều chỉnh trọng số.



Dể ngắn gọn, hãy định nghĩa lỗi của mạng là hàm .

Nếu ta mở rộng , ta thu được:

Nhớ rằng các giá trị duy nhất ta có thể điều chỉnh để thay đổi là các trọng số, . Trong phép tính vi phân, lấy đạo hàm là *tính tốc độ thay đổi của một hàm đối với một thay đổi vô cùng nhỏ trong một đối số đầu vào*. Trong trường hợp của chúng ta, nó có nghĩa là tính tốc độ thay đổi của hàm để đáp ứng một thay đổi rất nhỏ trong . Đó là những gì ta sẽ gọi để tính toán gradient, ta sẽ gọi là , tại một điểm trong bề mặt lỗi.

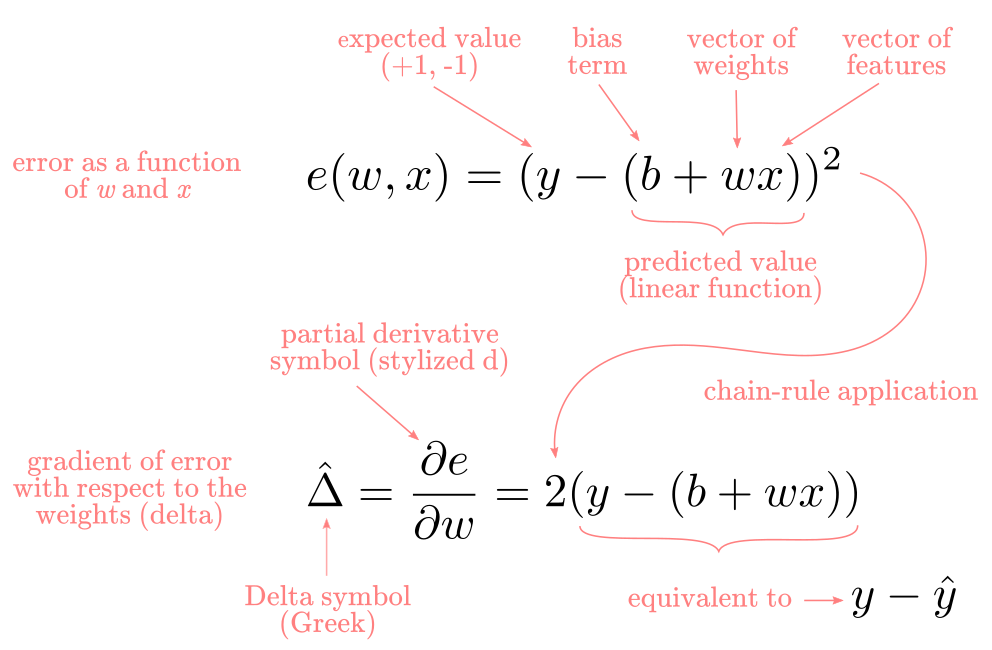
MSE:

Trong trường hợp của Adaline, chúng ta có thể định nghĩa hàm chi phí, J, để học trọng số dưới dạng tổng của bình phương sai số (SSE) giữa đầu ra tính toán và nhãn lớp thực tế:

1/2 chỉ được thêm vào để thuận tiện cho việc tính đạo hàm của hàm chi phí và hàm mất mát theo các tham số trọng số. Ưu điểm chính của hàm kích hoạt tuyến tính liên tục này, so với hàm bước đơn vị, là hàm chi phí trở nên khả vi. Một thuộc tính tuyệt vời khác của hàm chi phí này là hàm lồi; do đó, chúng ta có thể sử dụng thuật toán tối ưu đơn giản nhưng mạnh mẽ gọi là gradient descent để tìm các trọng số tối thiểu hóa hàm chi phí.

Widrow và Hoff có một ý tưởng thay vì tính toán độ dốc cho tổng MSE , họ có thể tính gần đúng giá trị của gradient bằng cách tính toán đạo hàm riêng của sai số đối với trọng số mỗi lần lặp lại. Vì chúng ta xữ lí một trường hợp duy nhất, hàm đạo hàm trở thành:

Bằng cách áp dụng quy tắc chuỗi của phép tính, grandient của :



Độ dốc, trong trường hợp này, đơn giản **bằng 2 lần chênh lệch giửa giá trị mong đợi và giá trị dự đoán**. Giờ ta biết rằng cần cập nhật trọng số giửa các lần lặp. Cuối cùng, quy tắc cập nhật trọng số được phát biểu như sau: “Thay đổi trọng số bằng một portion của độ dốc âm ”. Chúng ta lấy độ dốc âm vì chúng ta muốn “xuống dốc”, nếu không thì ta sẽ đi sai hướng. Công thức như sau:

A picture containing text, font, diagram, handwriting

Description automatically generated

Chúng ta sử dụng portion của độ dốc thay vì giá trị đầy đủ để tránh “nảy xung quanh” (bouncing around) cực tiểu của hàm.

A picture containing text, diagram, line, plot

Description automatically generated

Trong phần bên trái quá lớn sẽ làm cho quả bóng nảy xung quan và ngăn cản nó đến cực tiểu. Trong phần bên phải, giá trị của đủ nhỏ để nó đạt đến cực tiểu sau vài lần lặp. Giá trị nhỏ giúp ta tìm ra cực tiểu nhưng cũng sẽ kéo dài thời gian training.