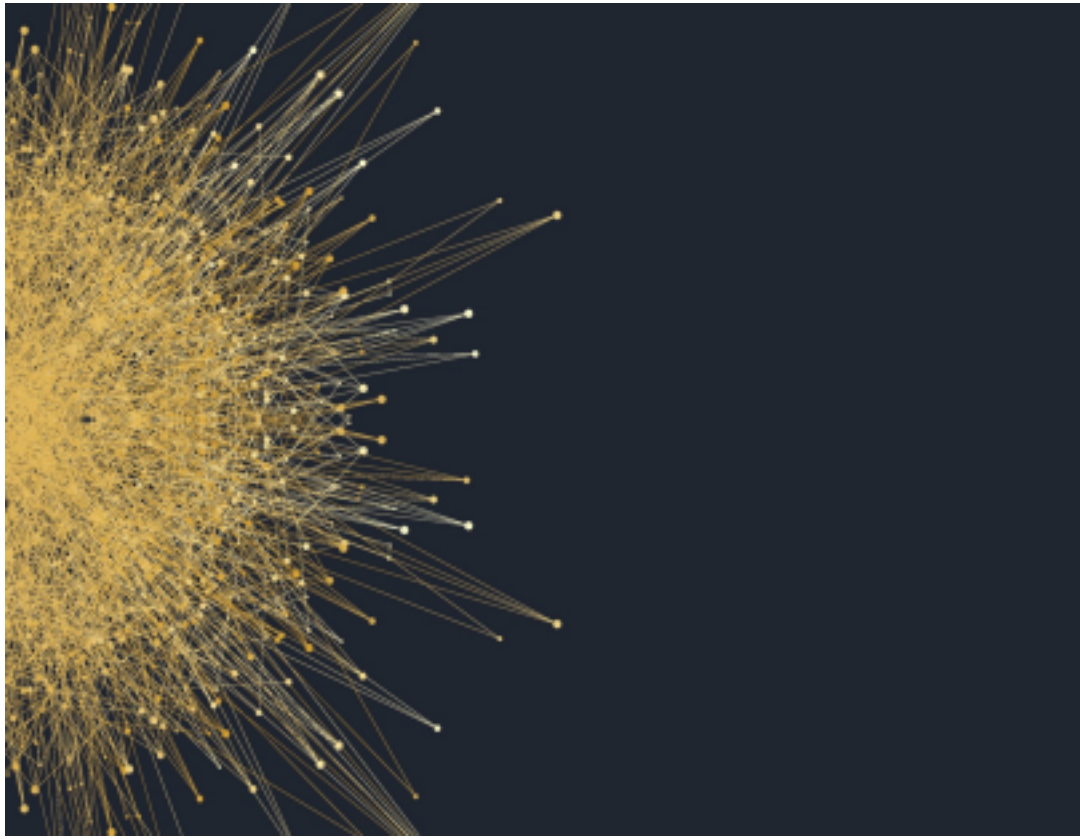
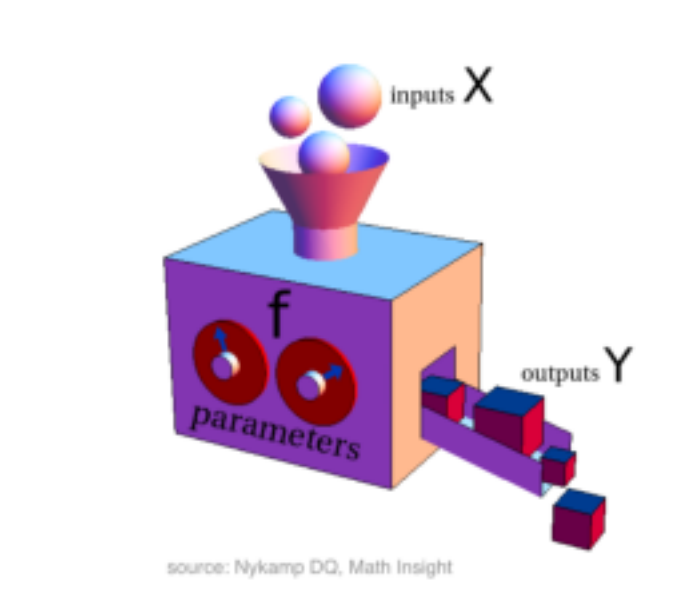
3. DL as hierarchical func finder:
3.0. Tóm tắt về linear model
3.1. 1 unit perceptron
3.2. 1 layer perceptron
3.3. XOR problem
3.4. 1 hidden layer perceptron
3.5. Why need deep?
3.6. Representation learning
3.7. multi-layer perceptrons
3.8. Backprop - deep func finder tool.
3.8. NN Variants: CNN, RNN.
(Practical case example across)



Tóm tắt về Linear model trong ML

"Cỗ máy tham số"



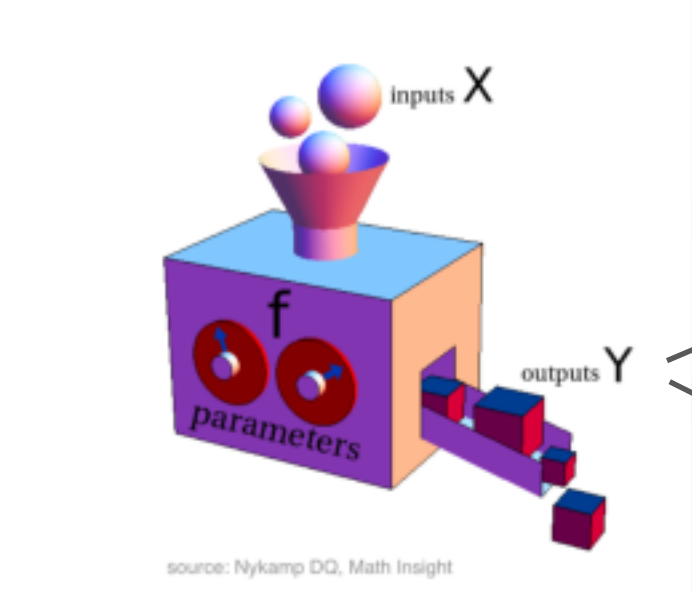
- Linear model sử dụng tham số để mô hình hoá bước mapping từ input ra output.

- Không nhất thiết phép mapping này cho ra kết quả hoàn toàn chính xác với thực tế, mà có thể xấp xỉ với kết quả thực tế (hay một phép mapping chuẩn).

- Chẳng hạn như hình bên, không nhất thiết đầu ra phải là 1 khối lập phương chuẩn mực, mà có thể có những lỗi nhỏ cũng không sao!

Tóm tắt về Linear model trong ML

"Cỗ máy tham số"

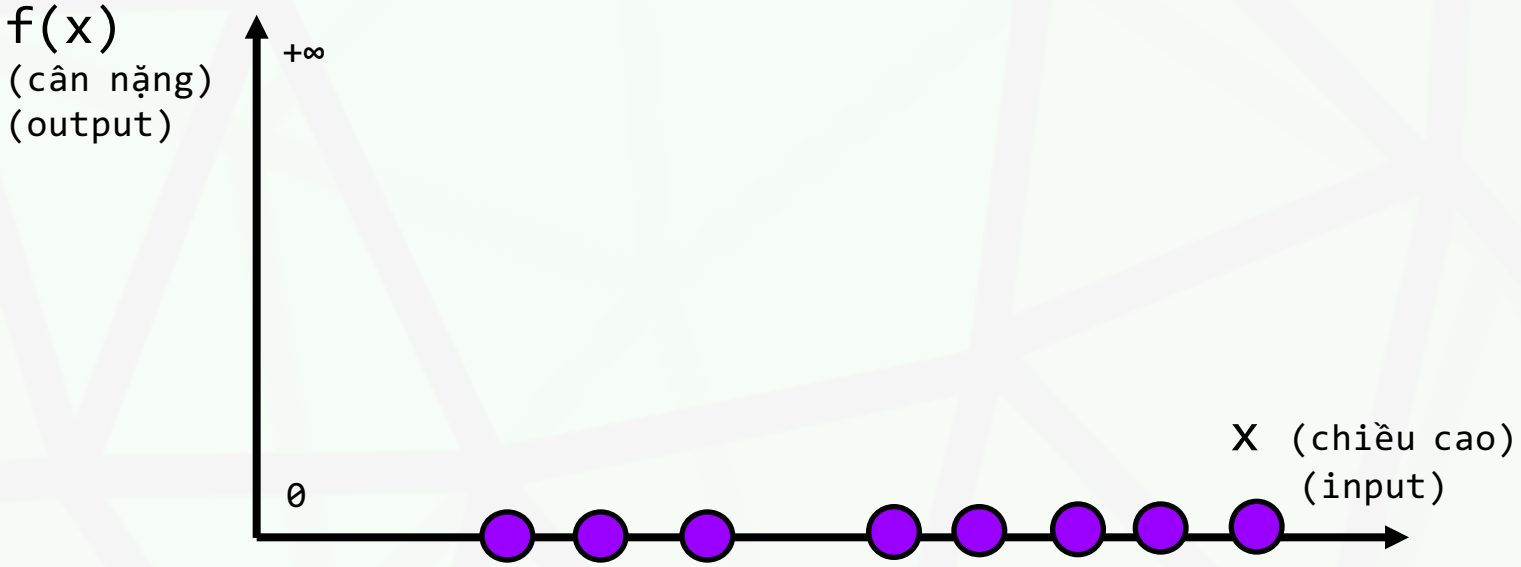


Bài toán Hồi quy

Bài toán Phân loại

Tóm tắt về Linear model trong ML

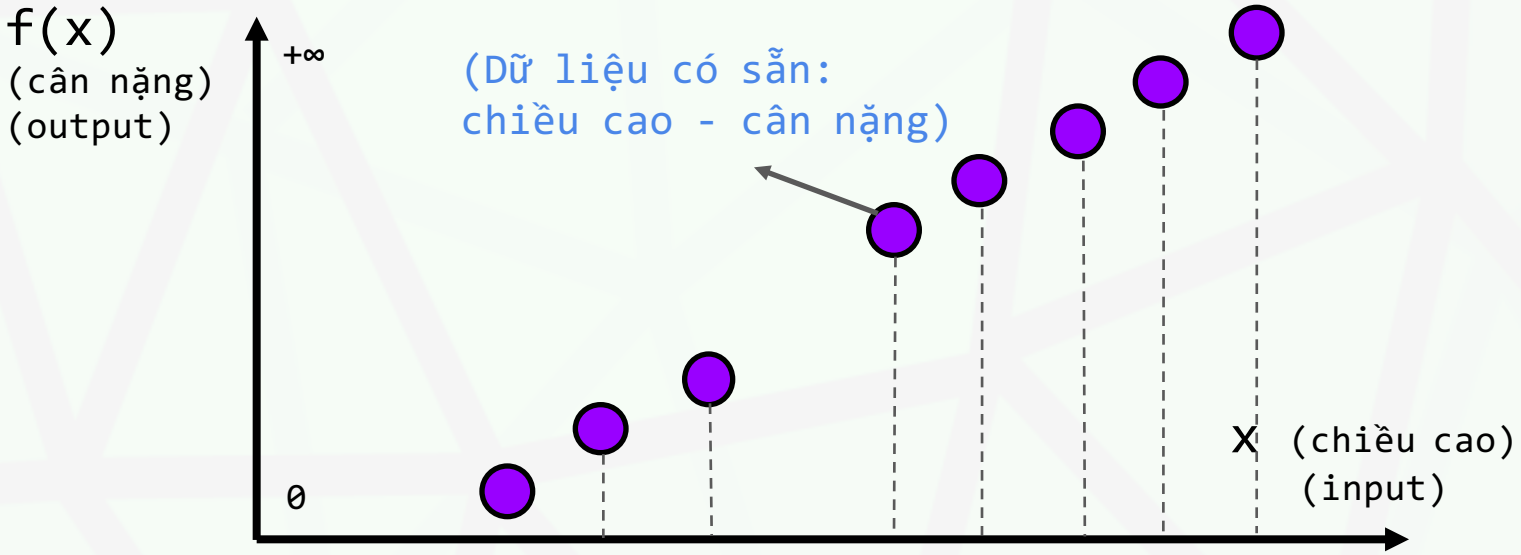
Ví dụ 1: Tính cân nặng của một người bình thường



- Chỉ số BMI của 1 người có sức khoẻ bình thường là **mối liên hệ** (pattern) giữa chiều cao và cân nặng của người đó, được tìm ra từ 1 tập ví dụ có sẵn.

Tóm tắt về Linear model trong ML

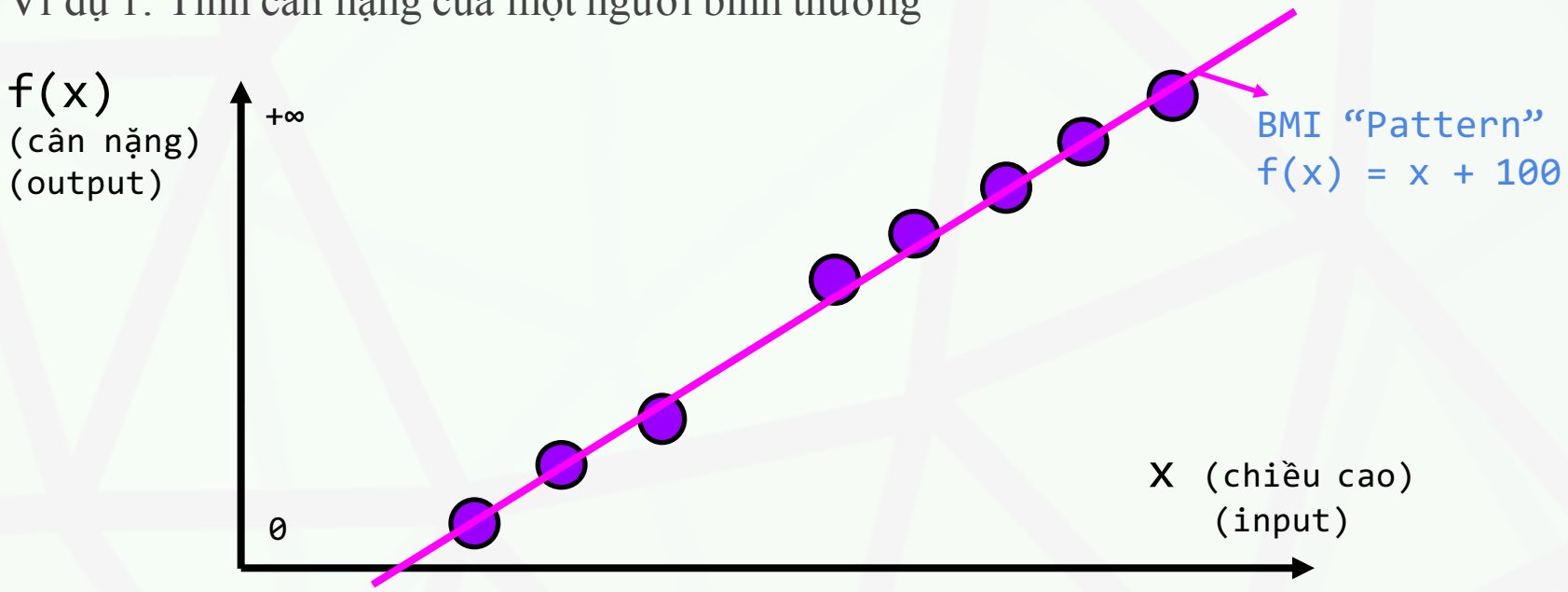
Ví dụ 1: Tính cân nặng của một người bình thường



- Chỉ số BMI của 1 người có sức khoẻ bình thường là **mối liên hệ** (pattern) giữa chiều cao và cân nặng của người đó, được tìm ra từ 1 tập ví dụ có sẵn.

Tóm tắt về Linear model trong ML

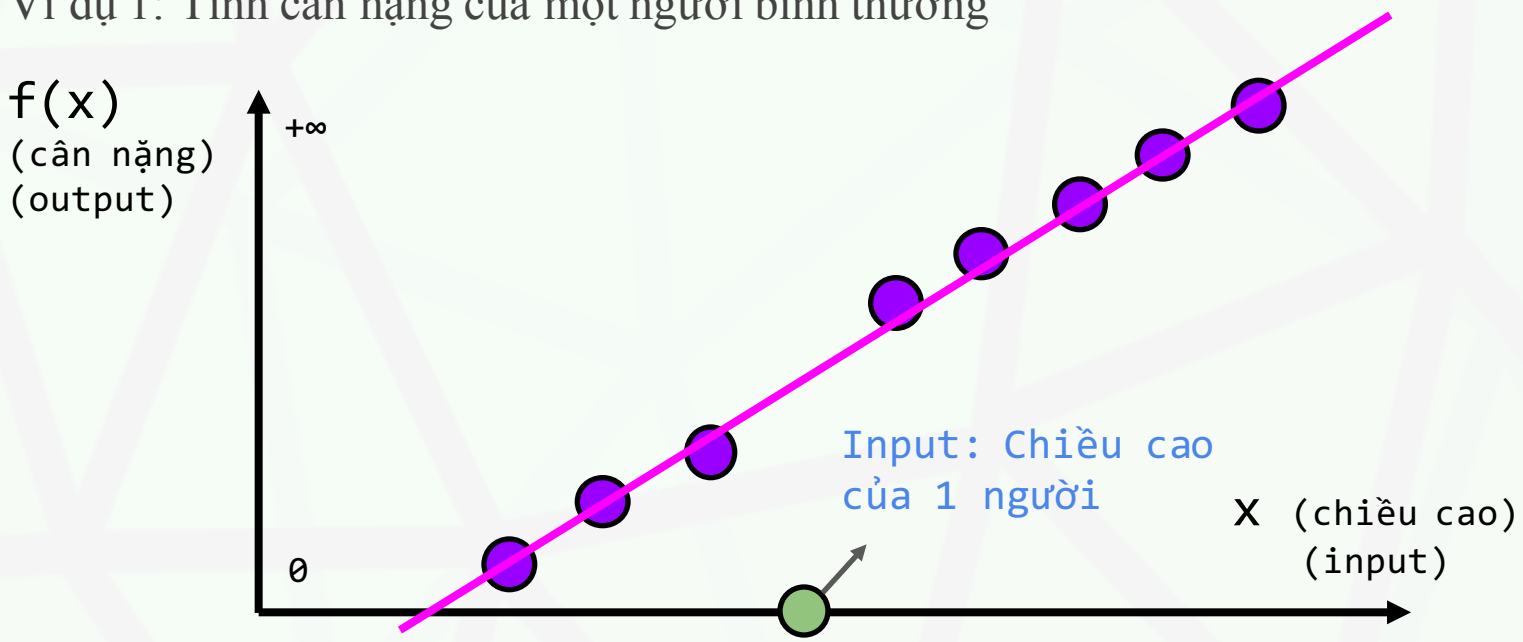
Ví dụ 1: Tính cân nặng của một người bình thường



- Chỉ số BMI của 1 người có sức khoẻ bình thường là **mối liên hệ** (pattern) giữa chiều cao và cân nặng của người đó, được tìm ra từ 1 tập ví dụ có sẵn.

Tóm tắt về Linear model trong ML

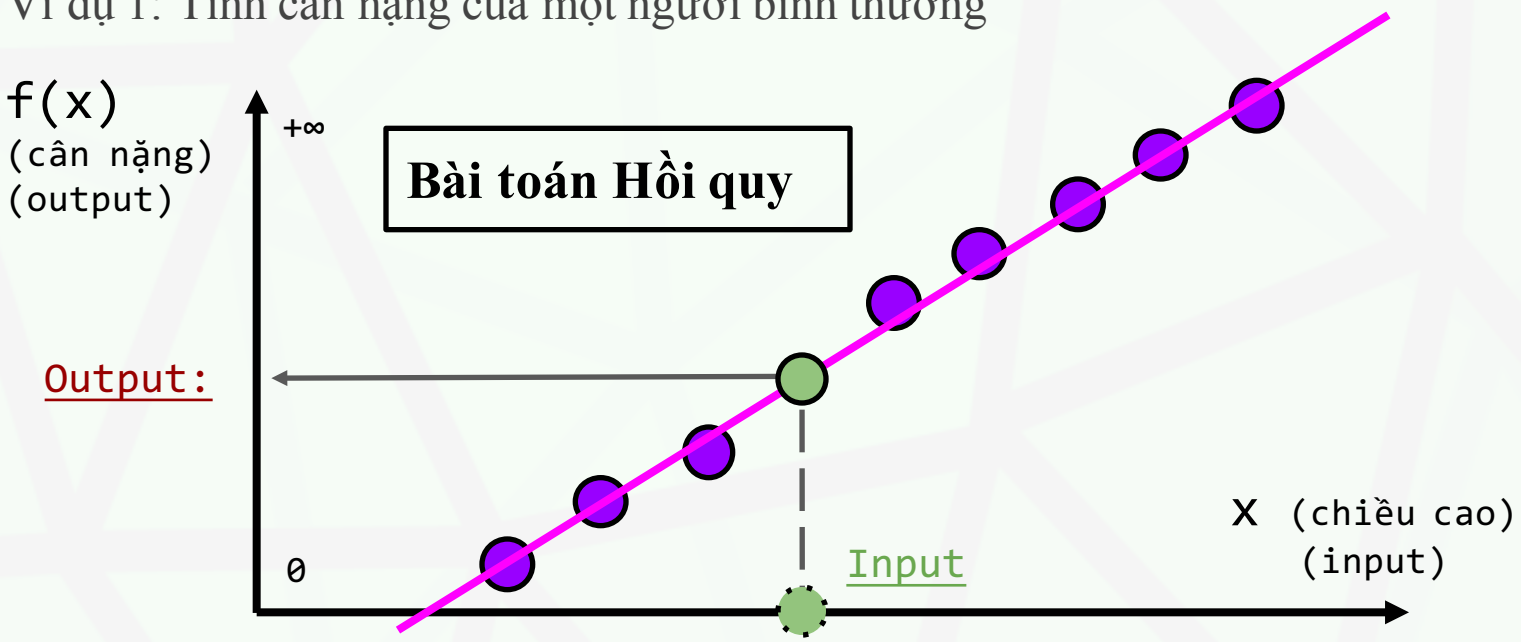
Ví dụ 1: Tính cân nặng của một người bình thường



- Chỉ số BMI của 1 người có sức khoẻ bình thường là **mối liên hệ** (pattern) giữa chiều cao và cân nặng của người đó, được tìm ra từ 1 tập ví dụ có sẵn.
- Khi biết chiều cao của một người (có vóc dáng bình thường), dựa trên chỉ số BMI của người bth, ta có thể dự đoán **số cân nặng** của người đó.

Tóm tắt về Linear model trong ML

Ví dụ 1: Tính cân nặng của một người bình thường



- Chỉ số BMI của 1 người có sức khoẻ bình thường là **mối liên hệ** (pattern) giữa chiều cao và cân nặng của người đó, được tìm ra từ 1 tập ví dụ có sẵn.
- Khi biết chiều cao của một người (có vóc dáng bình thường), dựa trên chỉ số BMI của người bth, ta có thể dự đoán **số cân nặng** của người đó.

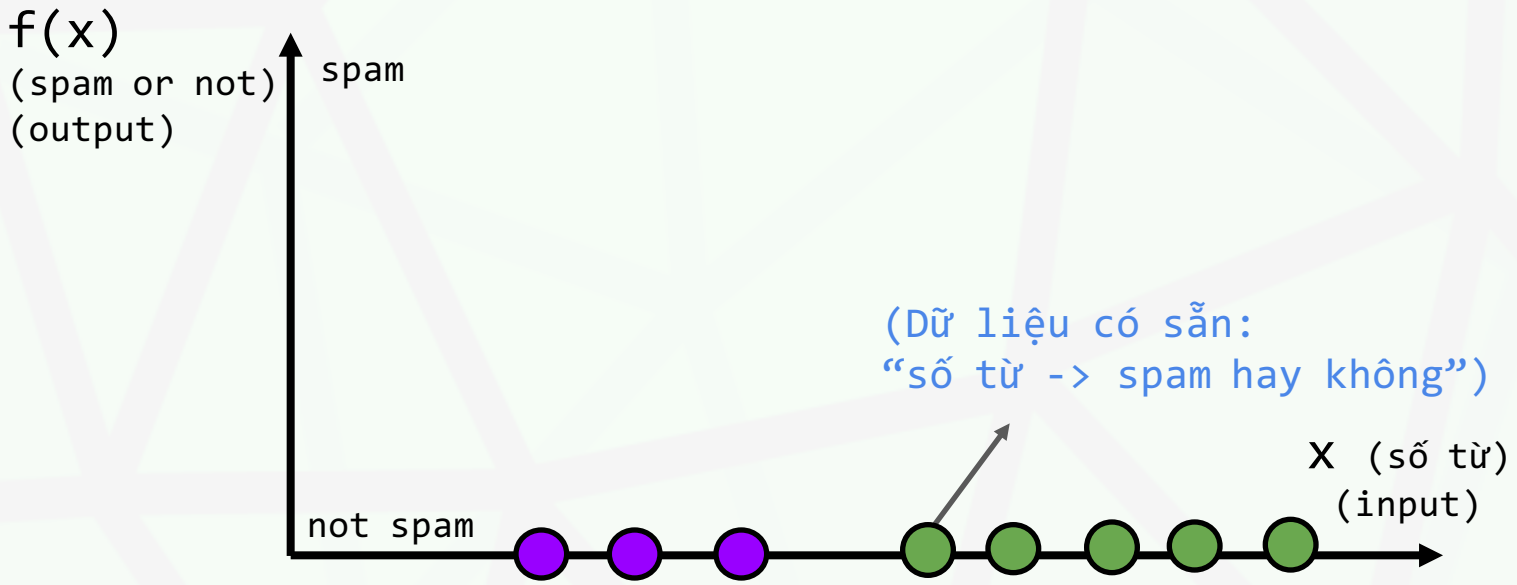
Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email

- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email

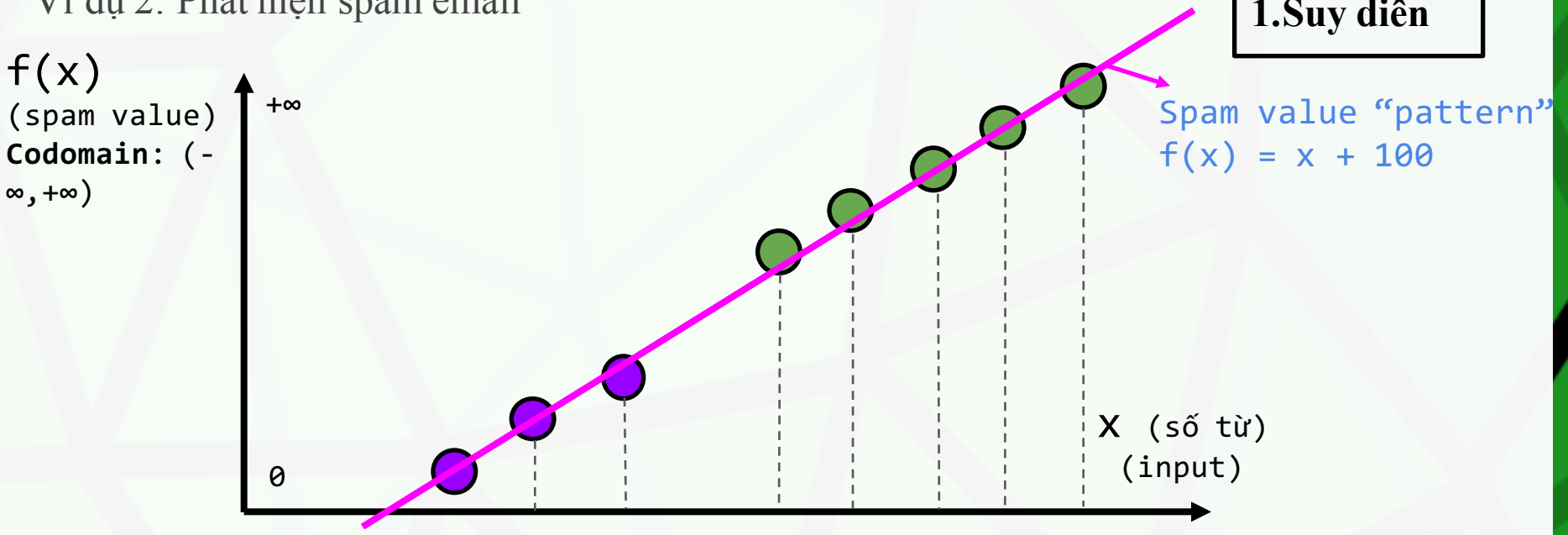


- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email

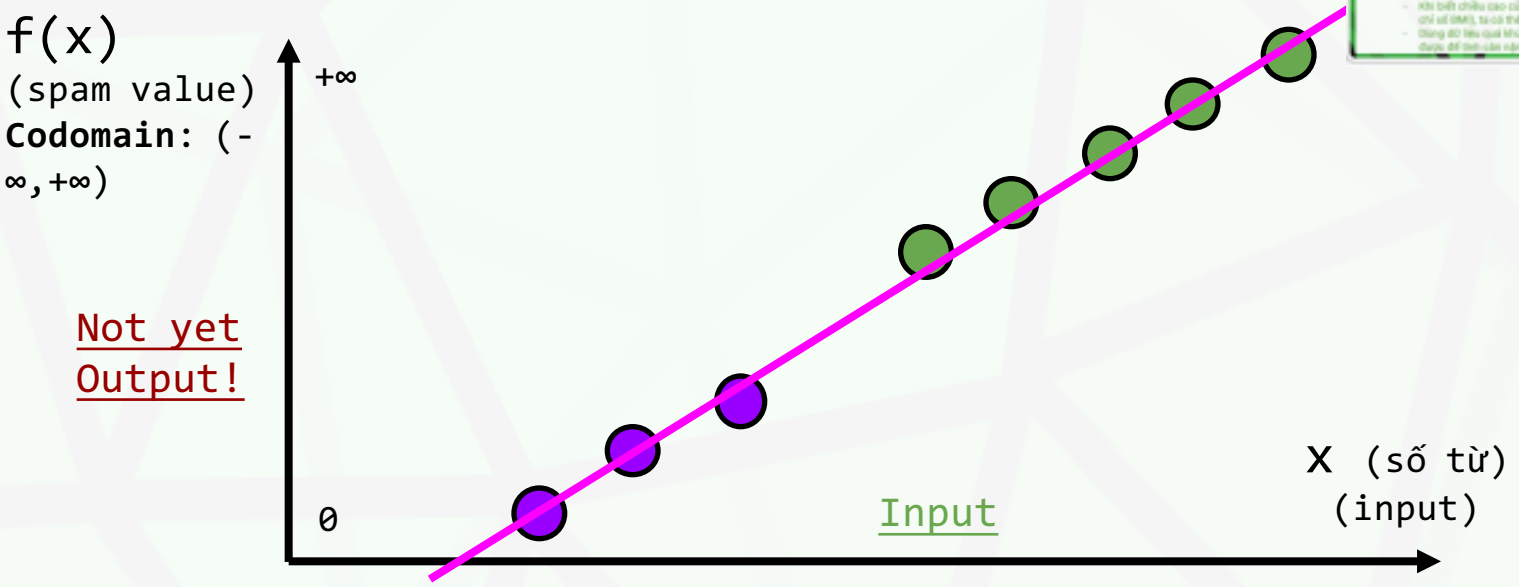
1.Suy diễn



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

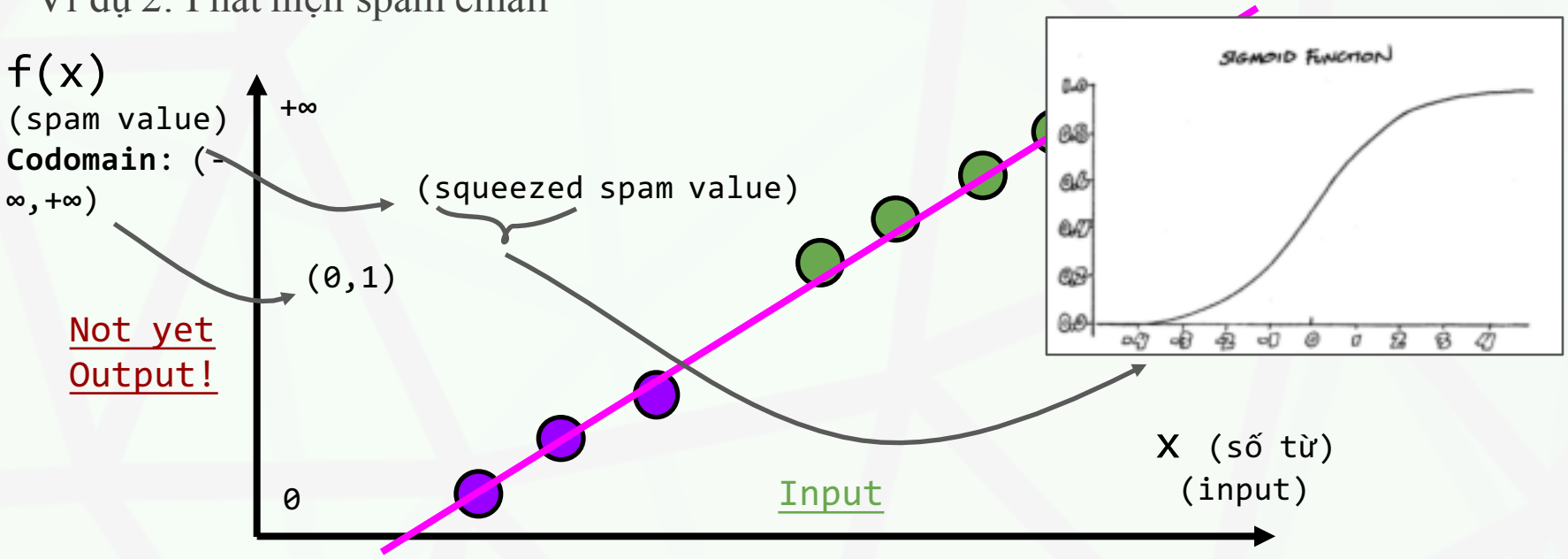
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

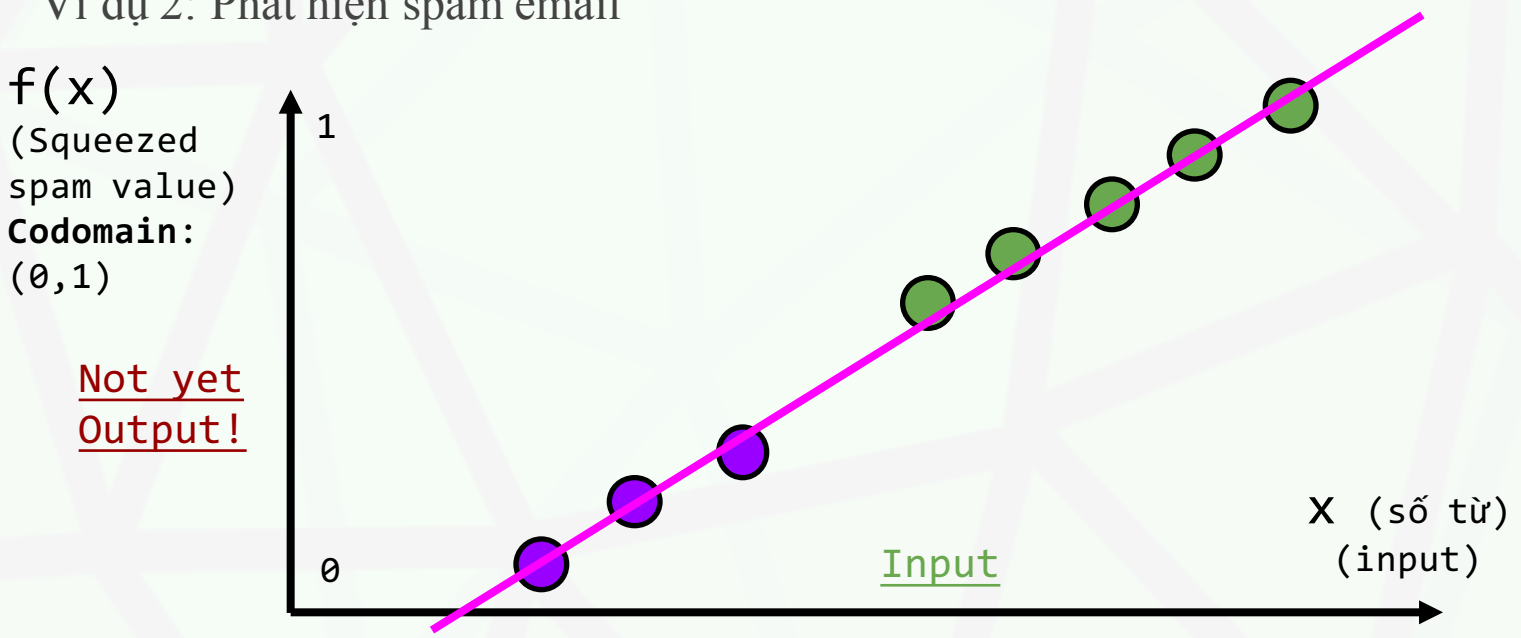
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

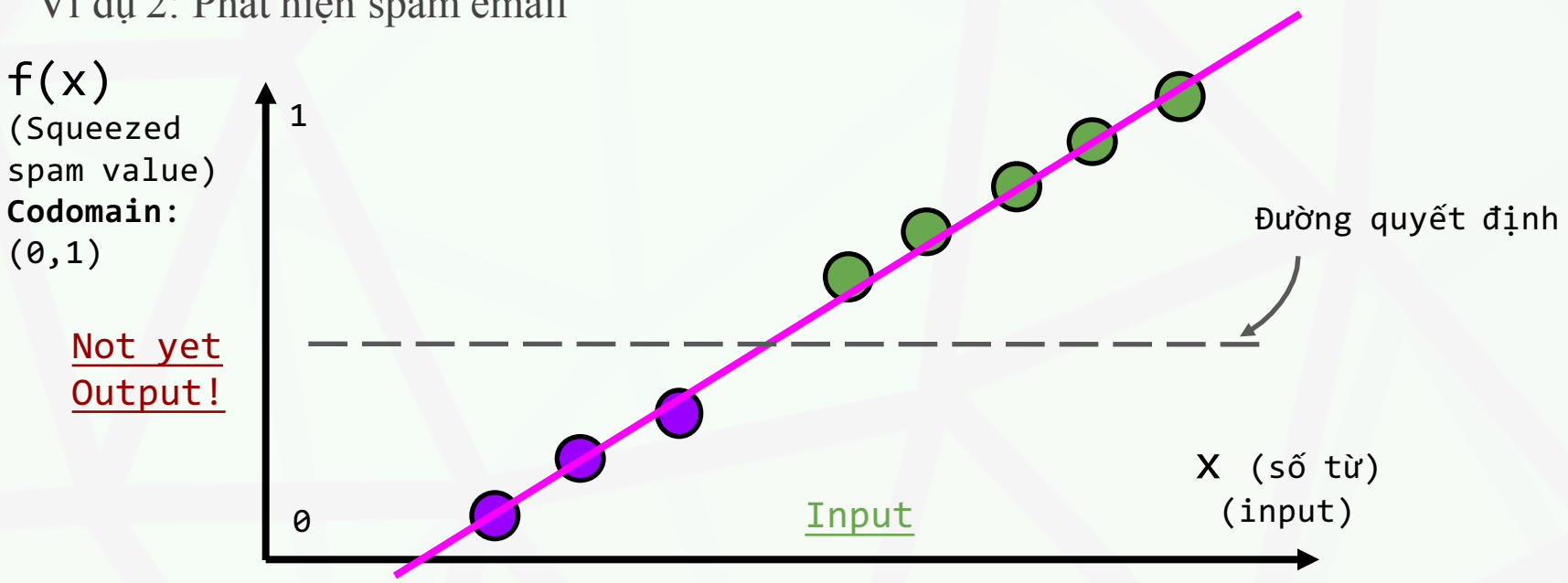
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email

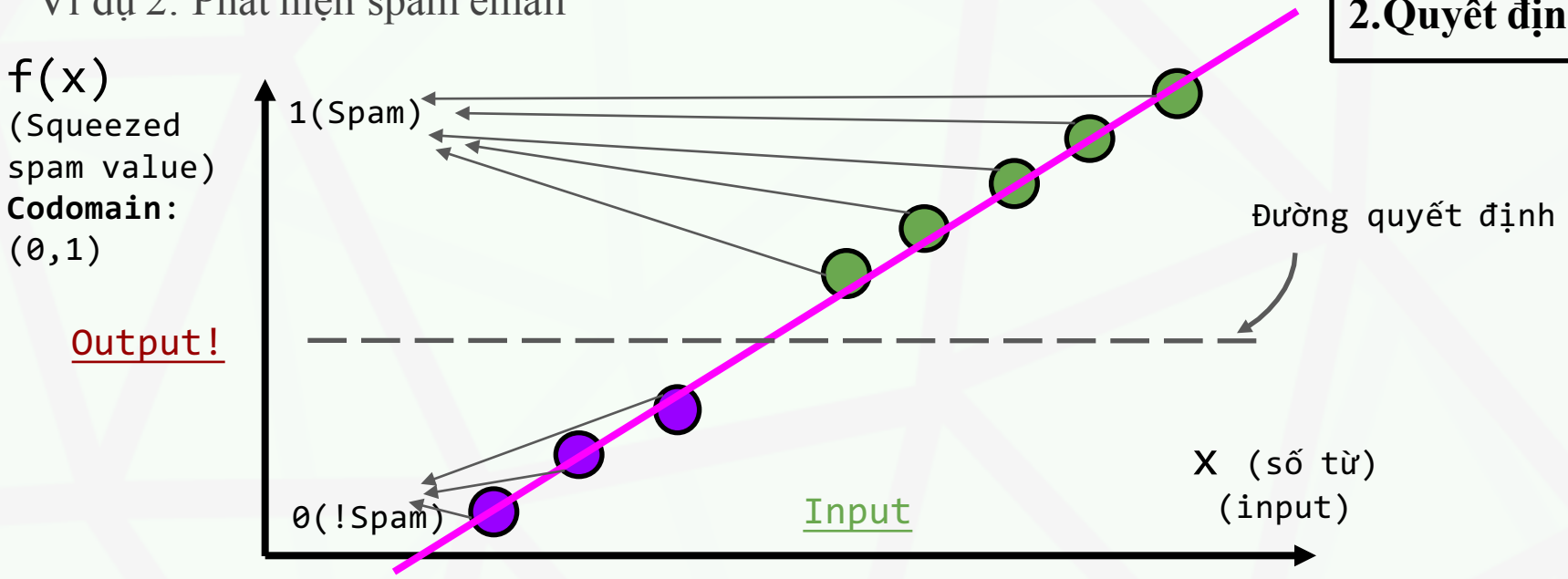


- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email

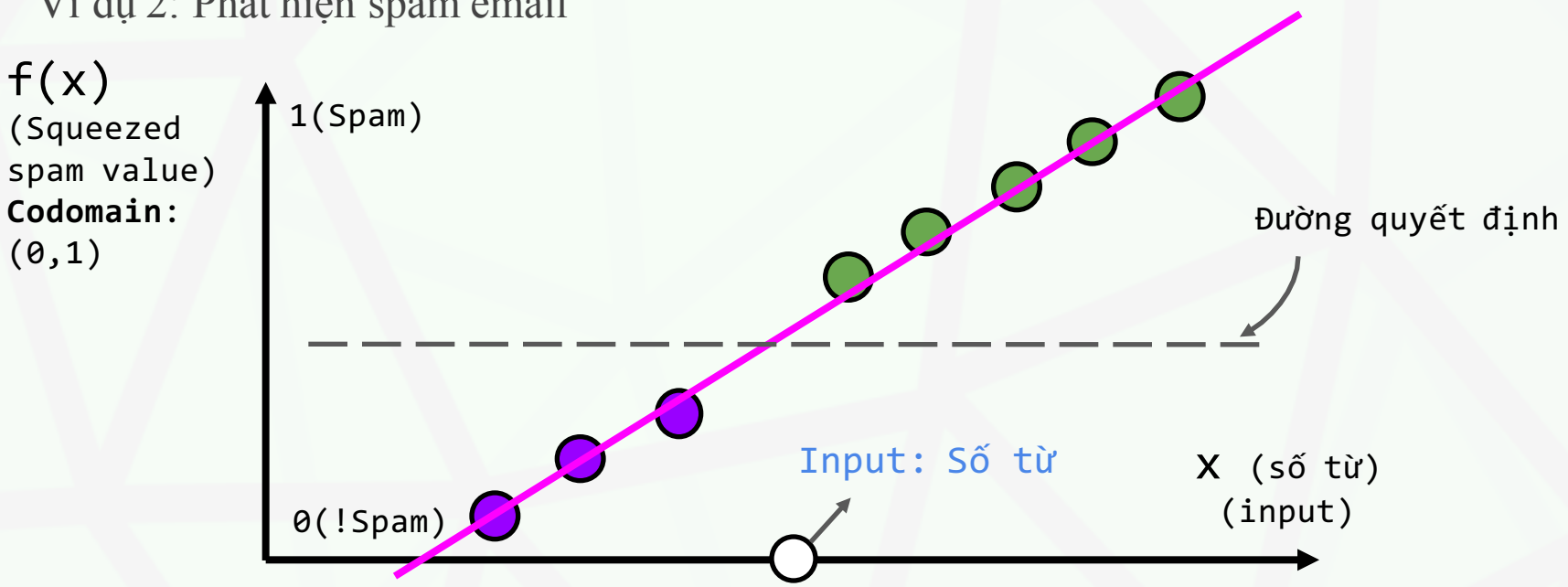
2.Quyết định



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

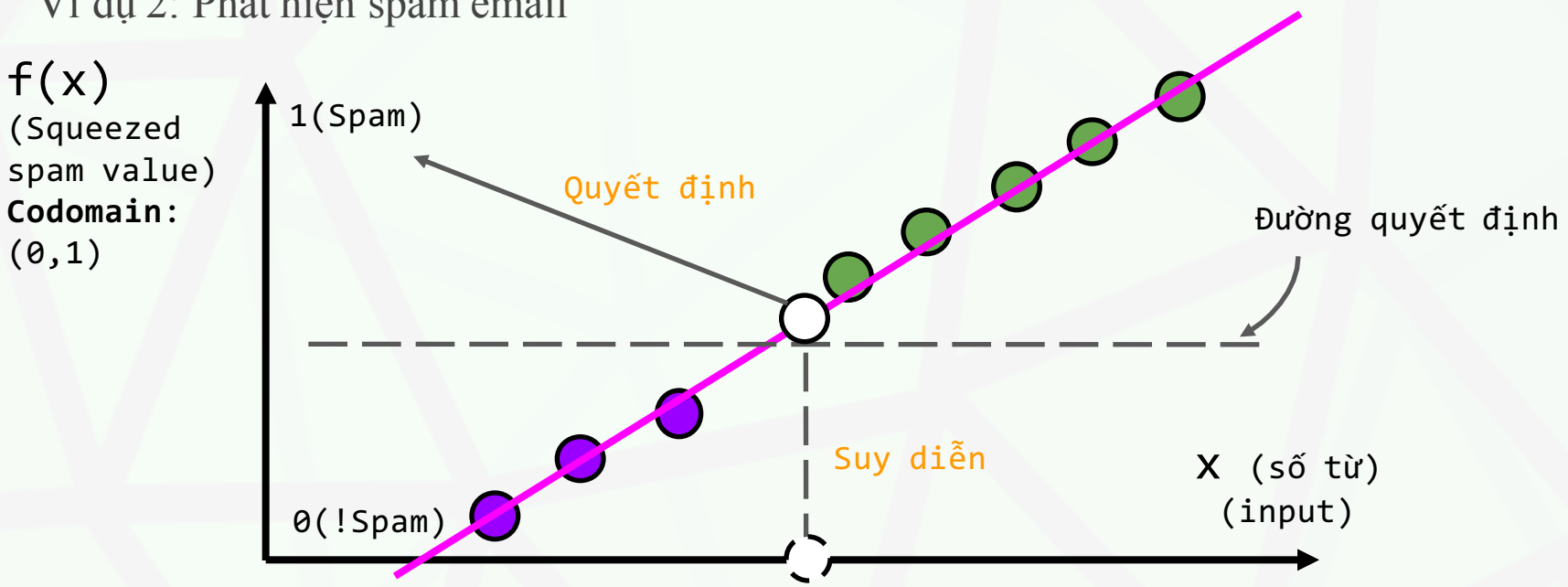
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

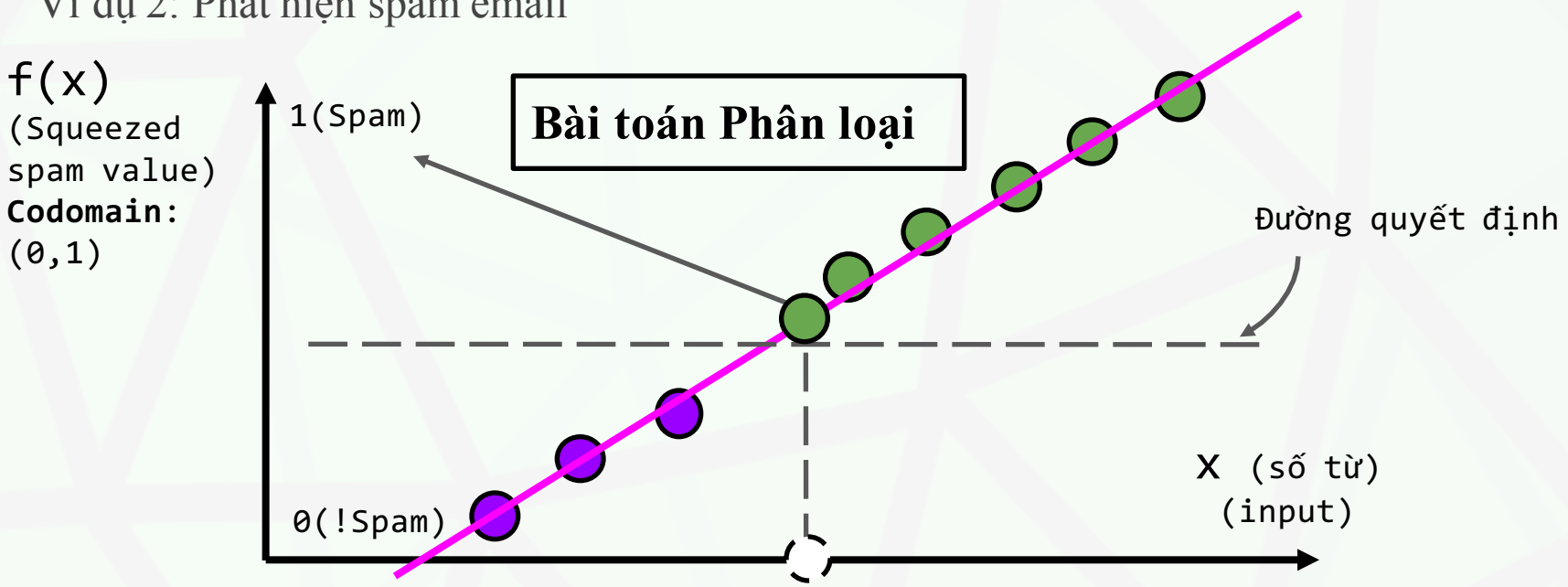
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

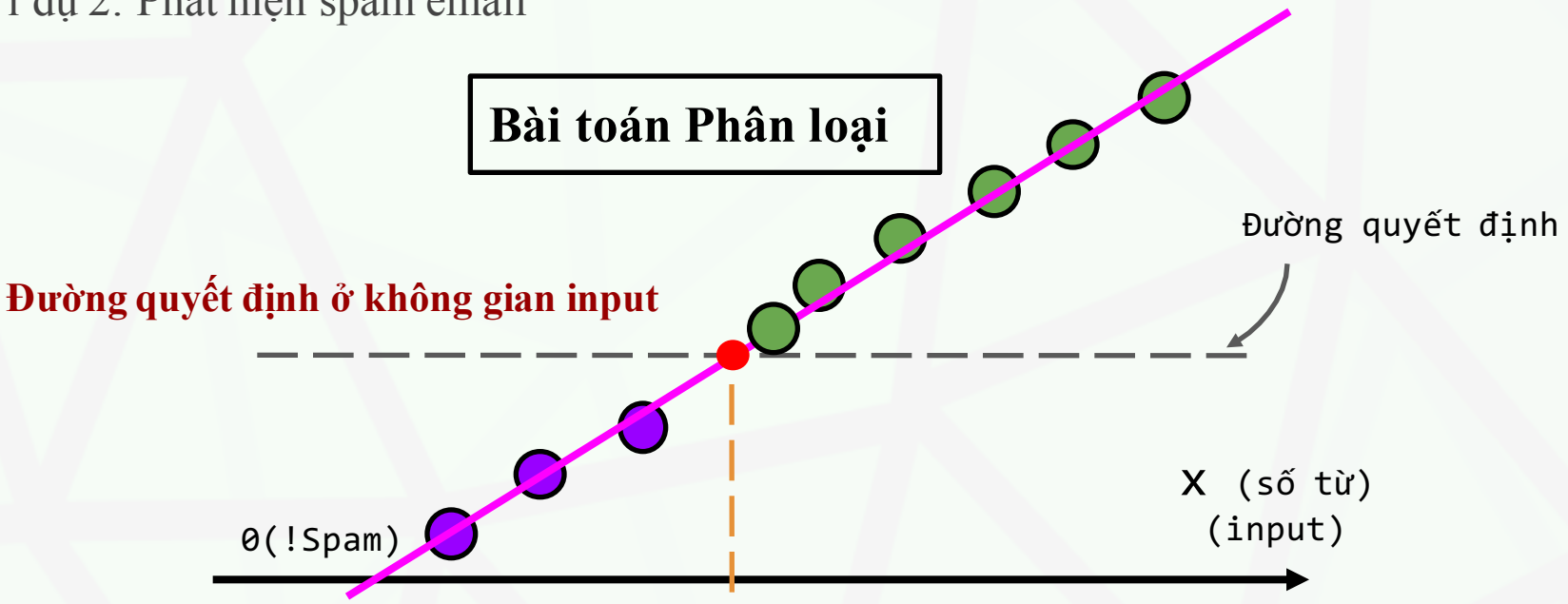
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

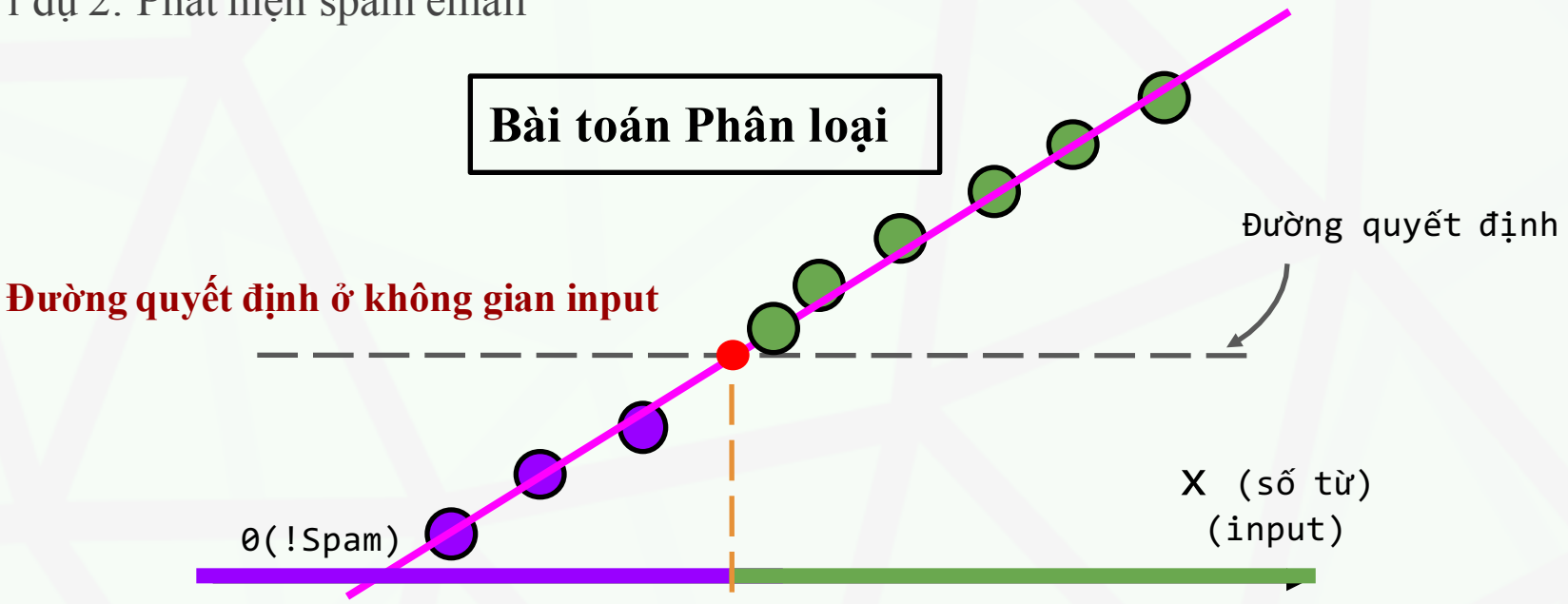
Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

Tóm tắt về Linear model trong ML

Ví dụ 2: Phát hiện spam email



- Dựa trên số lượng từ "giảm giá" trong email, ta có thể dự đoán email đó là **spam hay không**.
- Bài toán này gồm 2 bước: **Suy diễn** và **Quyết định**

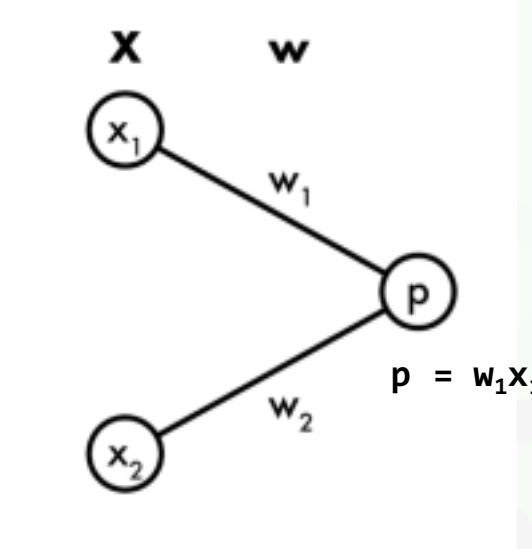
Neural Network

- Dạng cơ bản nhất của NN là một **Linear model**, output là **tổ hợp tuyến tính** của các input.

- Mạng **Deep NN** là dạng mở rộng của NN, gồm **Linear model (I)** với **hàm phi tuyến (II)** và **cấu trúc phân cấp (III)** (*hierarchical structure*)

Neural Network

Một đơn vị



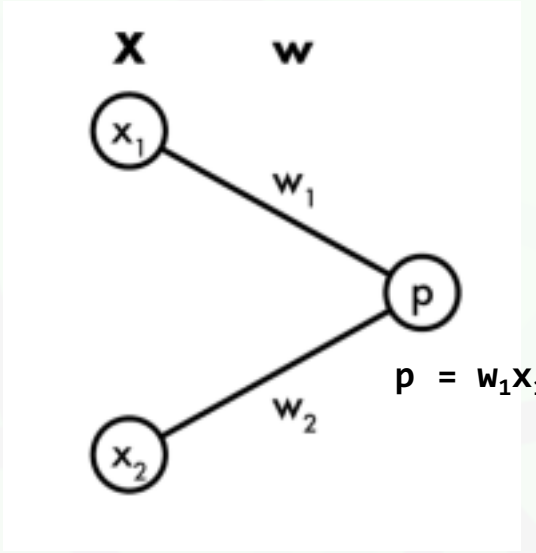
$$p = w_1x_1 + w_2x_2 + b$$

- Dạng cơ bản nhất của NN là một **Linear model (Perceptron)**, output là **tổ hợp tuyến tính** của các input.

- Mạng **Deep NN** là dạng mở rộng của NN, gồm **Linear model** với **hàm phi tuyến** và **cấu trúc phân cấp** (*hierarchical structure*)

Neural Network

Một đơn vị



$$p = w_1x_1 + w_2x_2 + b$$

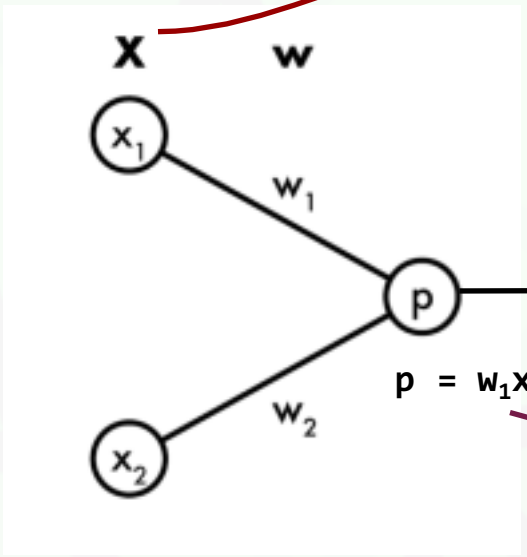
- Dạng cơ bản nhất của NN là một **Linear model**, output là **tổ hợp tuyến tính** của các input.

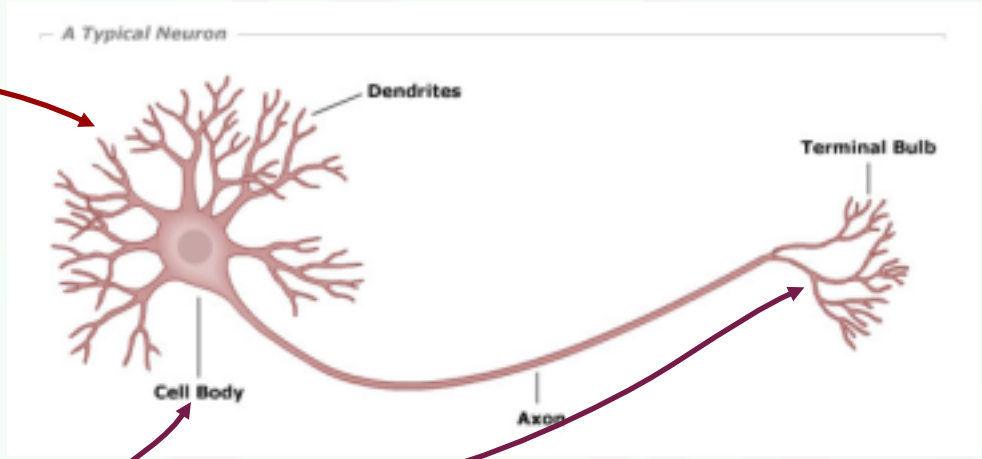
- Mạng **Deep NN** là dạng mở rộng của NN, gồm **Linear model** với **hàm phi tuyến** và **cấu trúc phân cấp** (*hierarchical structure*)

Liên quan gì tới Nơ-ron thần kinh của con người ?

Neural Network

Một đơn vị

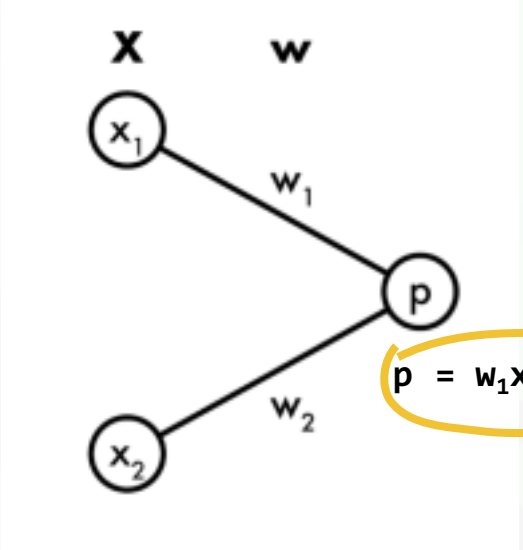




Liên quan gì tới Nơ-ron thần kinh của con người ?

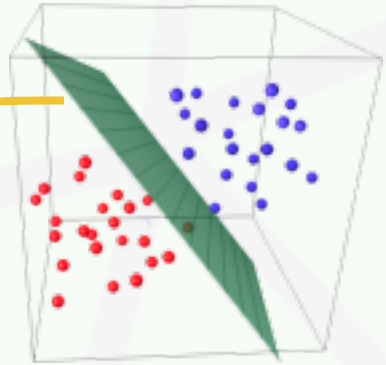
Neural Network

Một đơn vị (mở rộng: 2 input features)

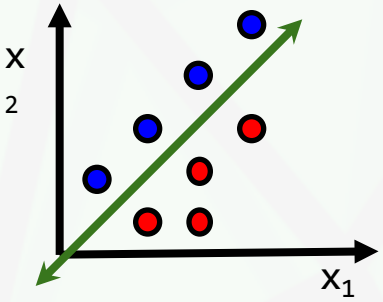


$$p = w_1x_1 + w_2x_2 + b$$

Bài toán Phân loại



Không gian input



http://playground.tensorflow.org/

Perceptron

Mô phỏng hoạt động của Neuron bằng hàm tuyến tính

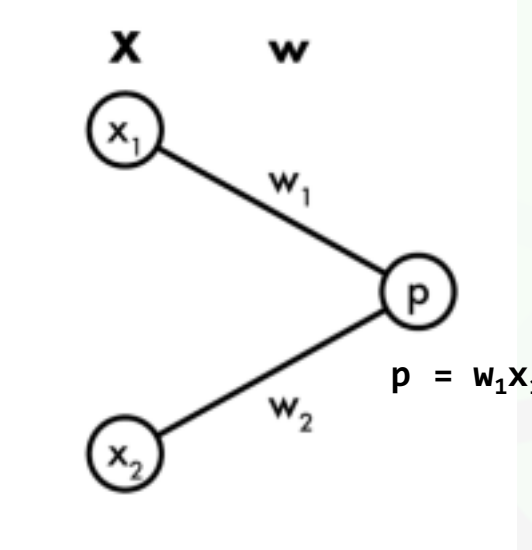
“Dễ” tìm tham số: ∃ thuật toán nhanh (Perceptron Algorithm) tìm tham số

Dễ giải thích

Tuy nhiên, khả năng biểu diễn giới hạn

Neural Network

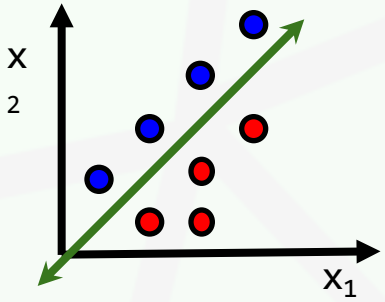
Một đơn vị (mở rộng: 2 input features)



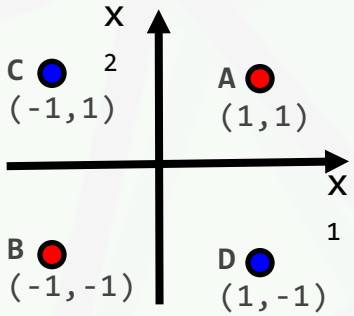
$$p = w_1x_1 + w_2x_2 + b$$

Bài toán Phân loại

Easy!



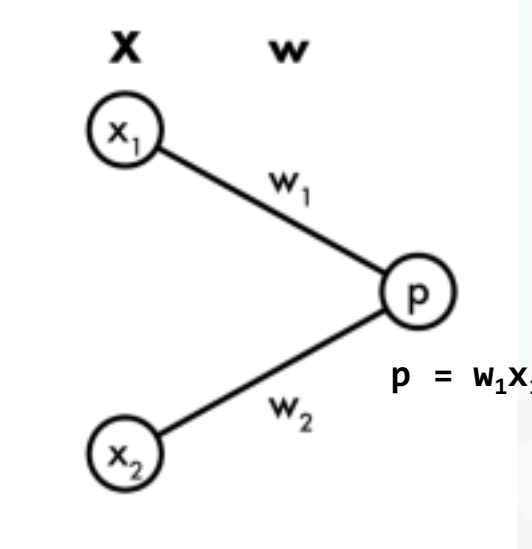
!Easy



- **Linear model (I)** cũng như hầu hết các ML models hoạt động tốt khi mối quan hệ giữa input features và output rõ ràng.

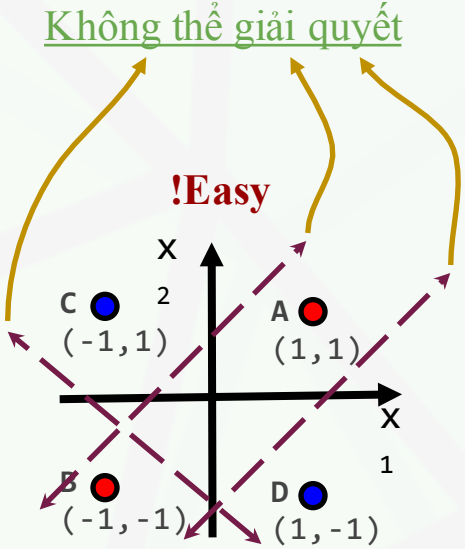
Neural Network

Một đơn vị (mở rộng: 2 input features)



$$p = w_1x_1 + w_2x_2 + b$$

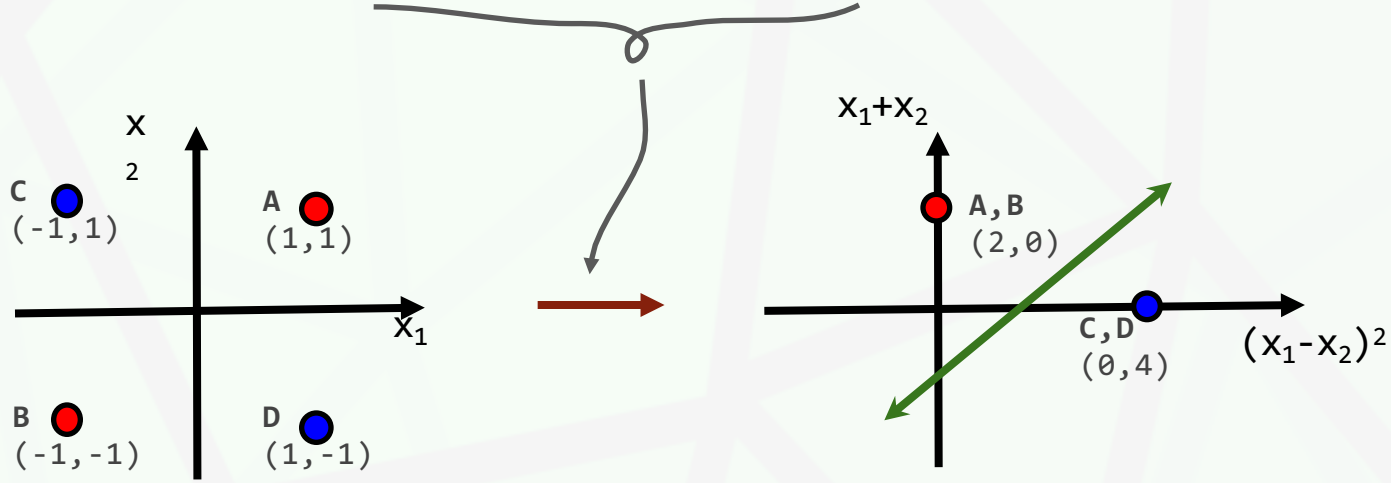
Bài toán Phân loại



- **Linear model (I)** cũng như hầu hết các ML models hoạt động tốt khi mối quan hệ giữa input features và output rõ ràng.

Neural Network

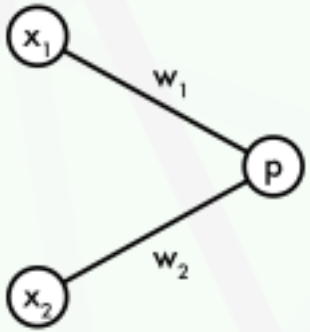
XOR: ML models sử dụng feature engineering



- Các kỹ sử ML thông thường phải tìm bằng cách thử-sai và lựa chọn các features phù hợp cho bài toán ML cụ thể.

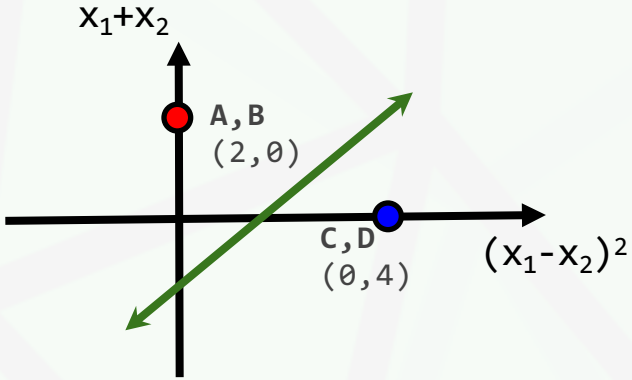
Neural Network

XOR: “automate” feature engineering



$p = f(w_1x_1 + w_2x_2)$

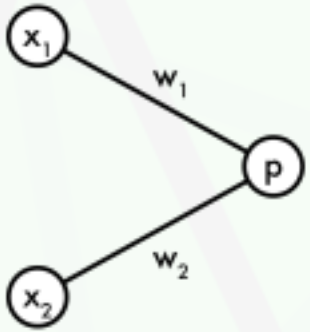
features mới



- Cơ bản, bước feature engineering sử dụng tổ hợp tuyến tính của input features, sau đó sử dụng 1 hàm nào đó để biến đổi tổ hợp này: $f(w_1x_1+w_2x_2)$ để tạo được các features mong muốn.
- Tại sao ta không học luôn quá trình extract features này?

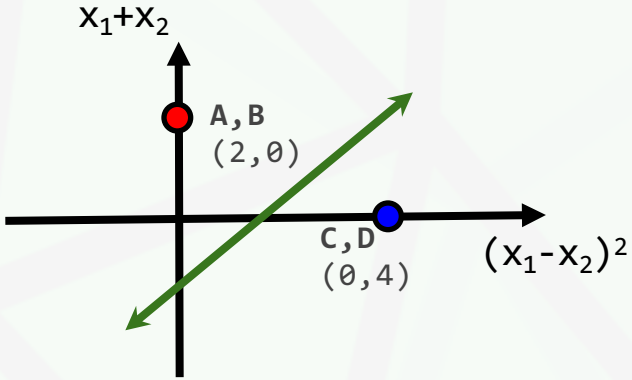
Neural Network

XOR: “automate” feature engineering



$p = f(w_1x_1 + w_2x_2)$

features mới

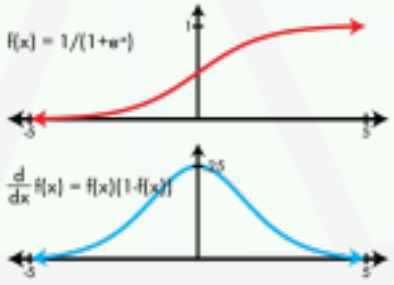


- Cơ bản, bước feature engineering sử dụng tổ hợp tuyến tính của input features, sau đó sử dụng 1 hàm nào đó để biến đổi tổ hợp này: $f(w_1x_1+w_2x_2)$ để tạo được các features mong muốn.
- Tại sao ta không học luôn quá trình extract features này? -> NN với **hàm phi tuyến**

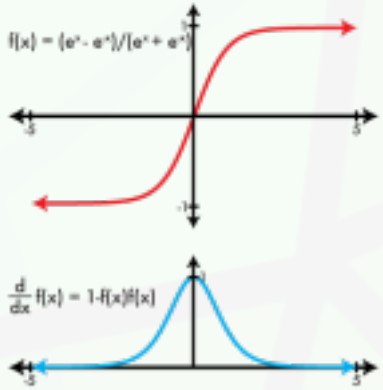
Neural Network

Hàm phi tuyến (Non-linear/Activation function)

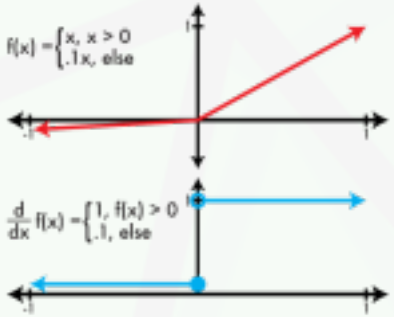
logistic



tanh



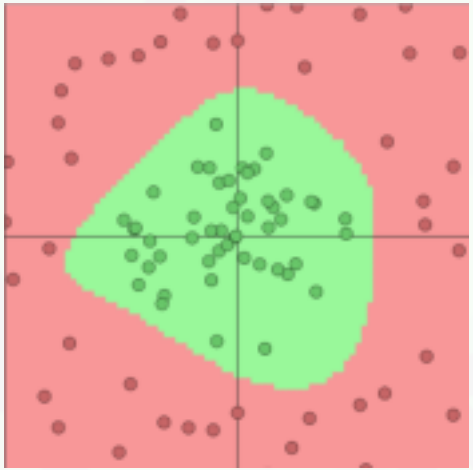
REctified Linear Unit (RELU)



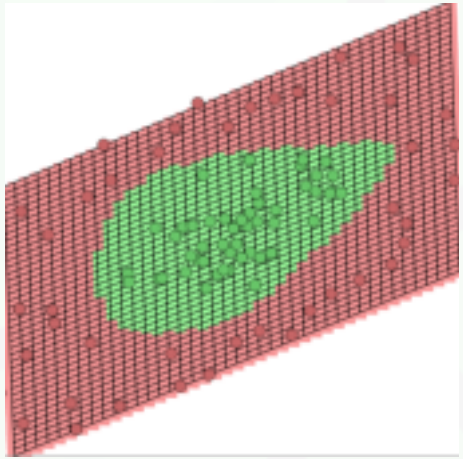
- Hàm phi tuyến giúp biến đổi không gian input features, tạo ra những features phù hợp giúp nhiệm vụ **Hồi quy/Phân loại** ở bước sau ở nên "straightforward"

Neural Network

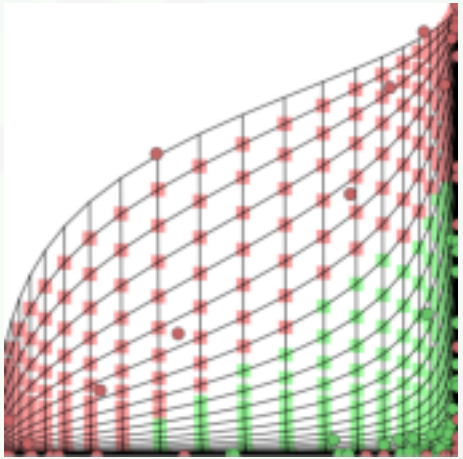
Hàm phi tuyến (Non-linear/Activation function) - ví dụ



W*
-->



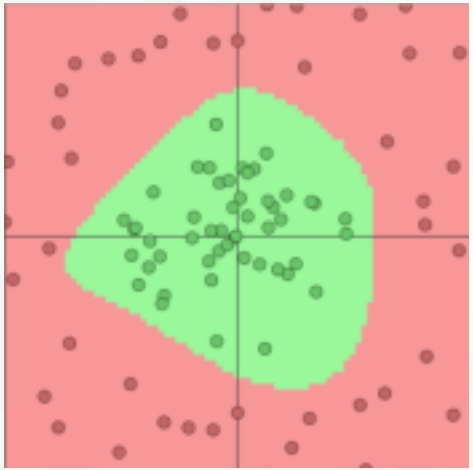
f()
-->



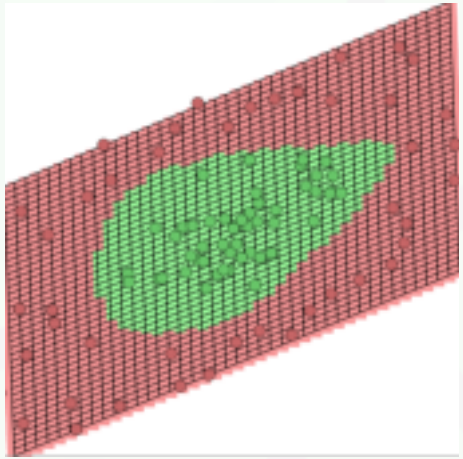
!Easy

Neural Network

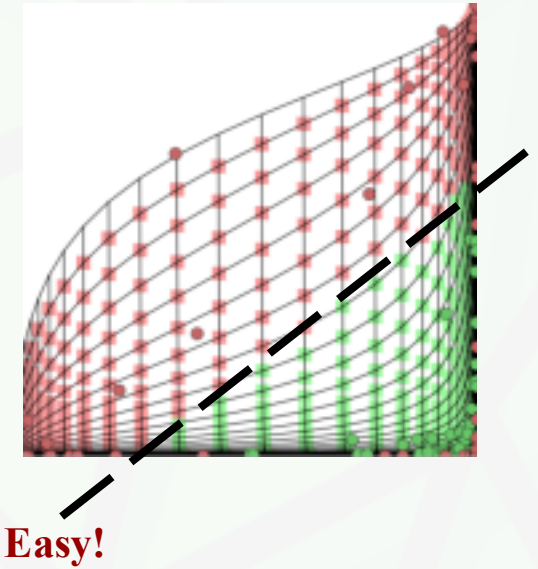
Hàm phi tuyến (Non-linear/Activation function) - ví dụ



W*
-->



f()
-->



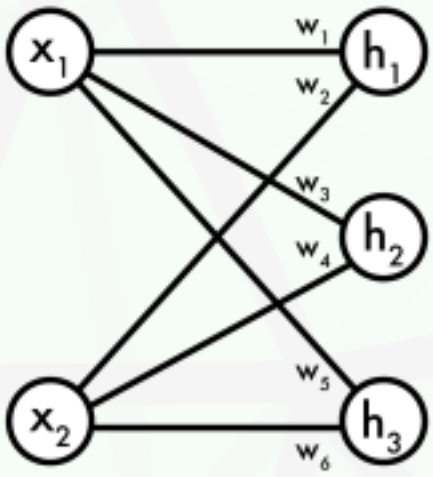
!Easy

Easy!

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Neural Network

Một lớp ẩn



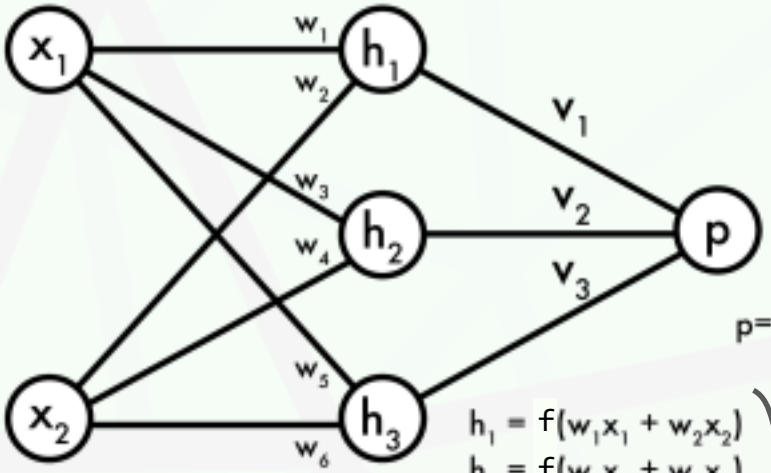
$$h_1 = f(w_1x_1 + w_2x_2)$$
$$h_2 = f(w_3x_1 + w_4x_2)$$
$$h_3 = f(w_5x_1 + w_6x_2)$$

Tạo feature mới

- Tạo ra feautures mới từ raw input, lớp này gọi là hidden layer *H*

Neural Network

Một lớp ẩn



$p = \varphi(v_1h_1 + v_2h_2 + v_3h_3)$ Squeezing function

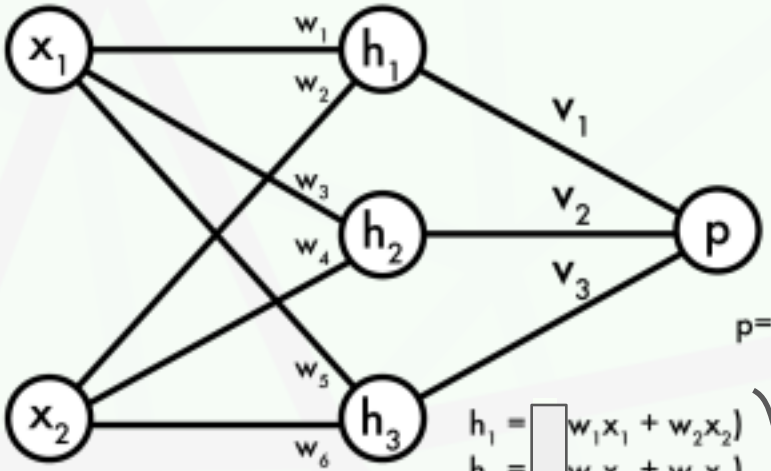
$h_1 = f(w_1x_1 + w_2x_2)$
$h_2 = f(w_3x_1 + w_4x_2)$
$h_3 = f(w_5x_1 + w_6x_2)$

Tạo feature mới

- Tạo ra feautures mới từ raw input, lớp này gọi là hidden layer *H*
- Các features này được sử dụng cho nhiệm vụ Hồi quy/Phân loại như ở các slide trước

Neural Network

Một lớp ẩn



$h_1 = [w_1x_1 + w_2x_2)$
$h_2 = [w_3x_1 + w_4x_2)$
$h_3 = [w_5x_1 + w_6x_2)$

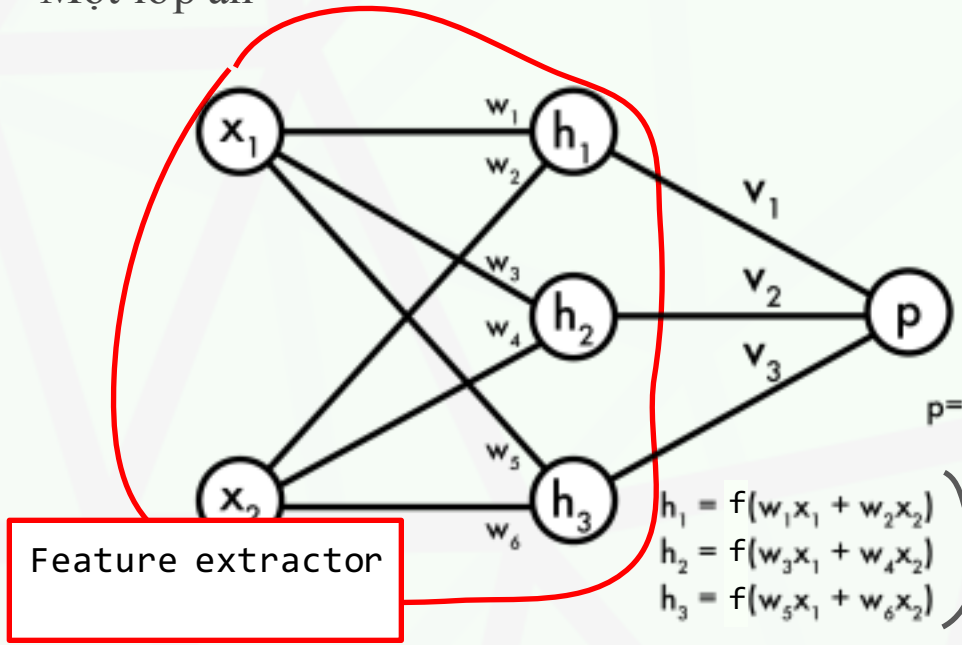
$p = \varphi(v_1h_1 + v_2h_2 + v_3h_3)$ Squeezing function

Lưu ý, nếu k có hàm f biến đổi phi tuyến
=> p = V*W*X = (V*W)*X
vẫn là Linear model!

- Tạo ra feautures mới từ raw input, lớp này gọi là hidden layer *H*
- Các features này được sử dụng cho nhiệm vụ Hồi quy/Phân loại như ở các slides trước

Neural Network

Một lớp ẩn



$p = \varphi(v_1h_1 + v_2h_2 + v_3h_3)$ Squeezing function

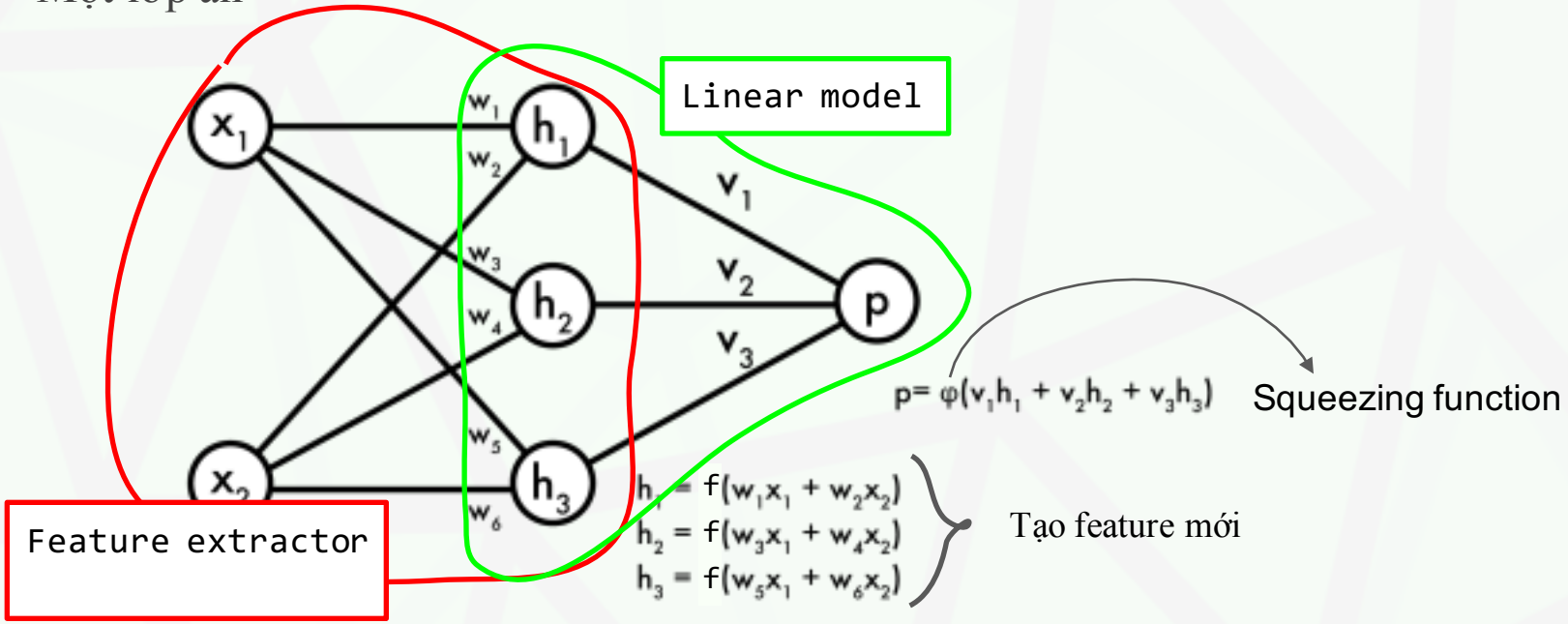
$h_1 = f(w_1x_1 + w_2x_2)$
$h_2 = f(w_3x_1 + w_4x_2)$
$h_3 = f(w_5x_1 + w_6x_2)$

Tạo feature mới

Công thức tạo Hidden layer: Linear model (I) + hàm phi tuyến (II)

Neural Network

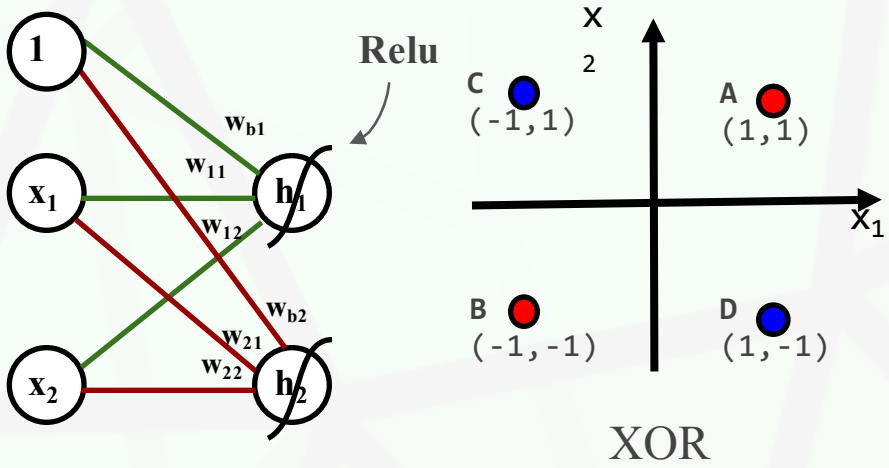
Một lớp ẩn



Công thức tạo Hidden layer: Linear model (I) + hàm phi tuyến (II)

Neural Network

Một lớp ẩn giải quyết XOR problem

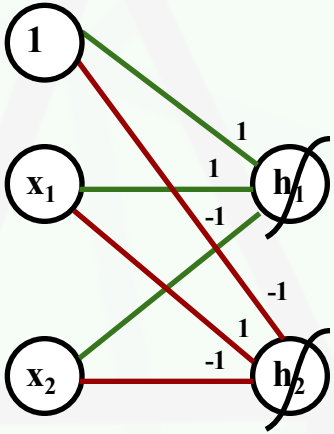


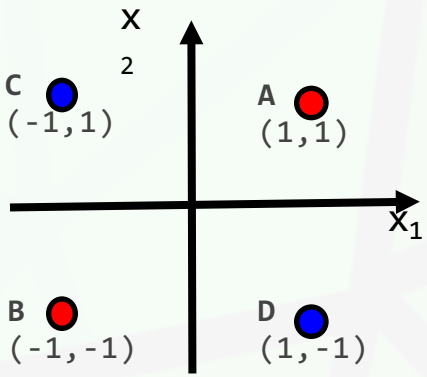
XOR

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

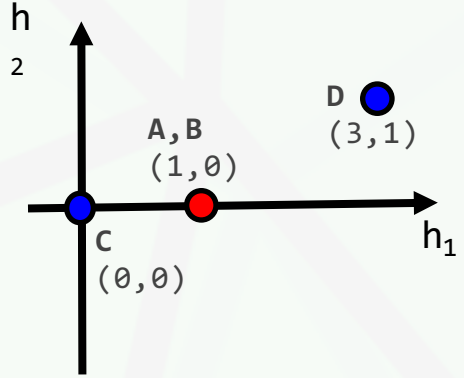
Neural Network

Một lớp ẩn giải quyết XOR problem





XOR

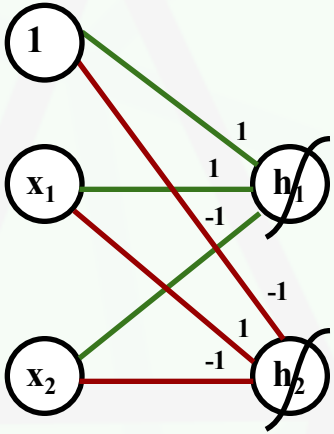


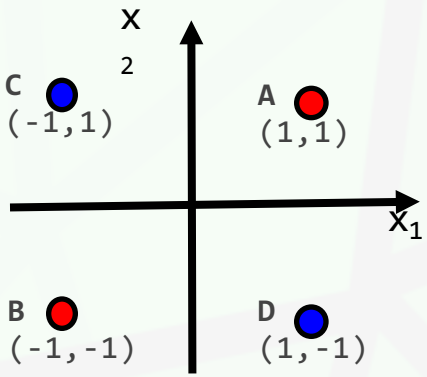
Transformed XOR

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

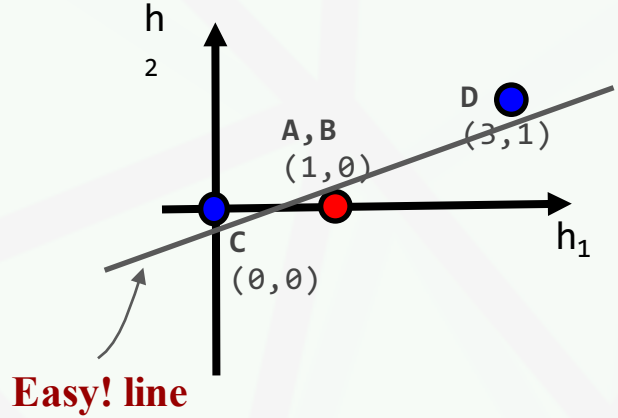
Neural Network

Một lớp ẩn giải quyết XOR problem





XOR



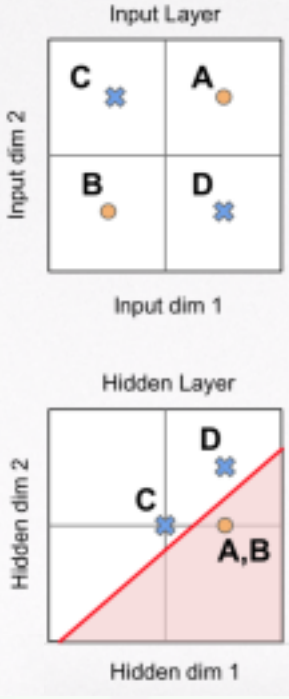
Transformed XOR

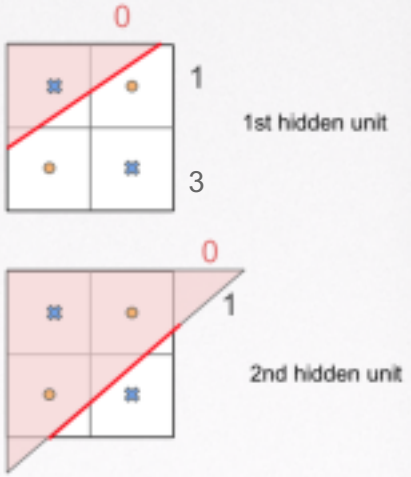
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Neural Network

Một lớp ẩn giải quyết XOR problem

Point	Original Space	Hidden Space	
A	(1,1)	(1,0)	Class 0
B	(-1,-1)	(1,0)	
C	(-1,1)	(0,0)	Class 1
D	(1,-1)	(3,1)	

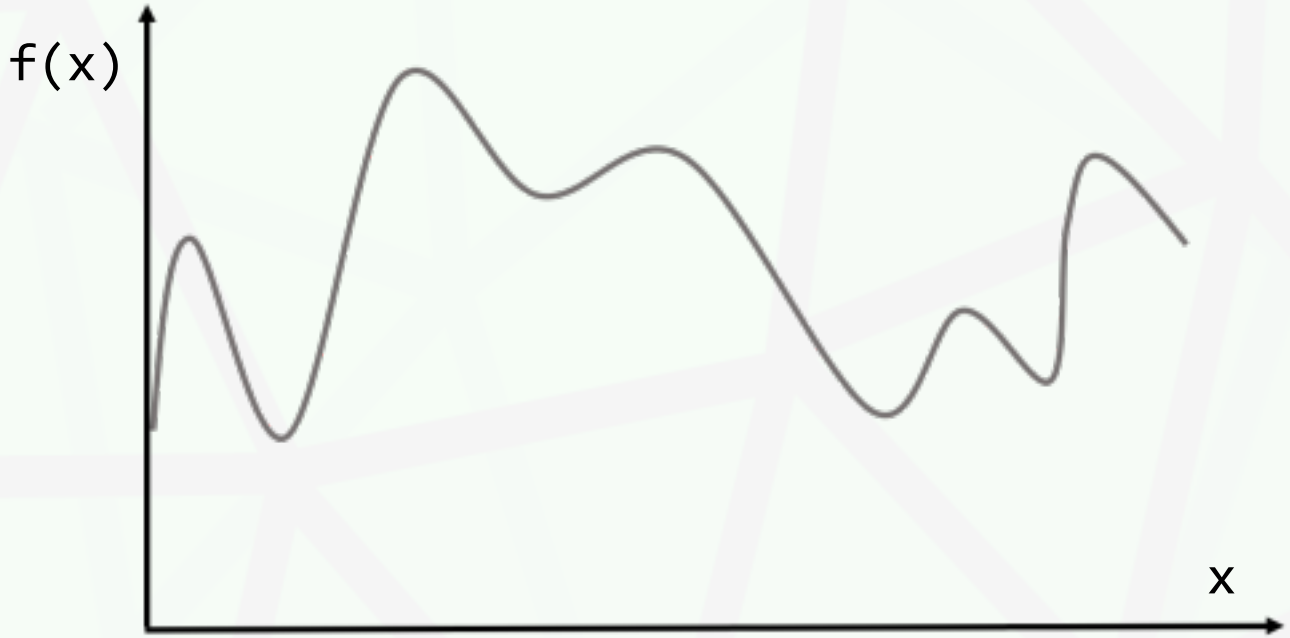




https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Neural Network

Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)

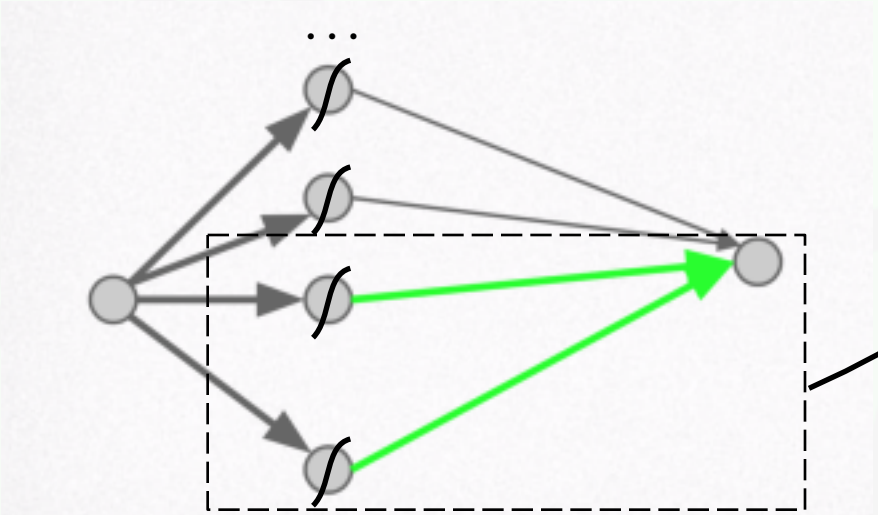


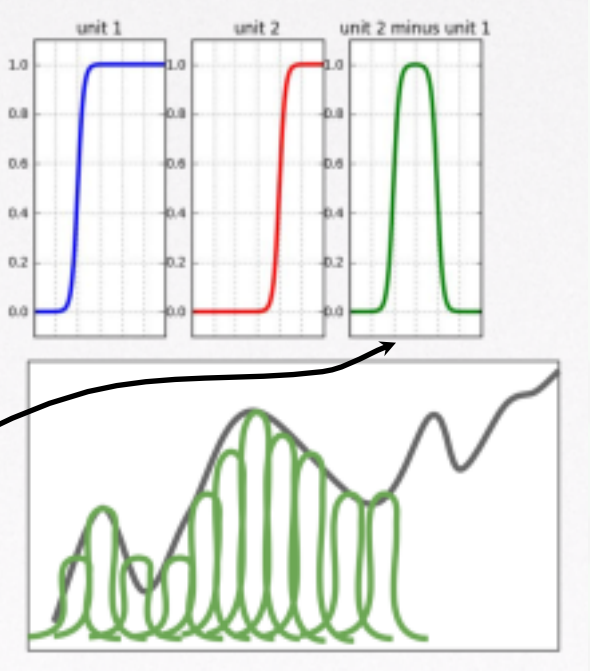
http://neuralnetworksanddeeplearning.com/chap4.html

Neural Network

Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)

- 1 cặp hidden units (với hàm sigmoid) có thể tạo thành 1 "bump".
- Tổ hợp các bumps có thể tạo thành 1 hàm số bất kì

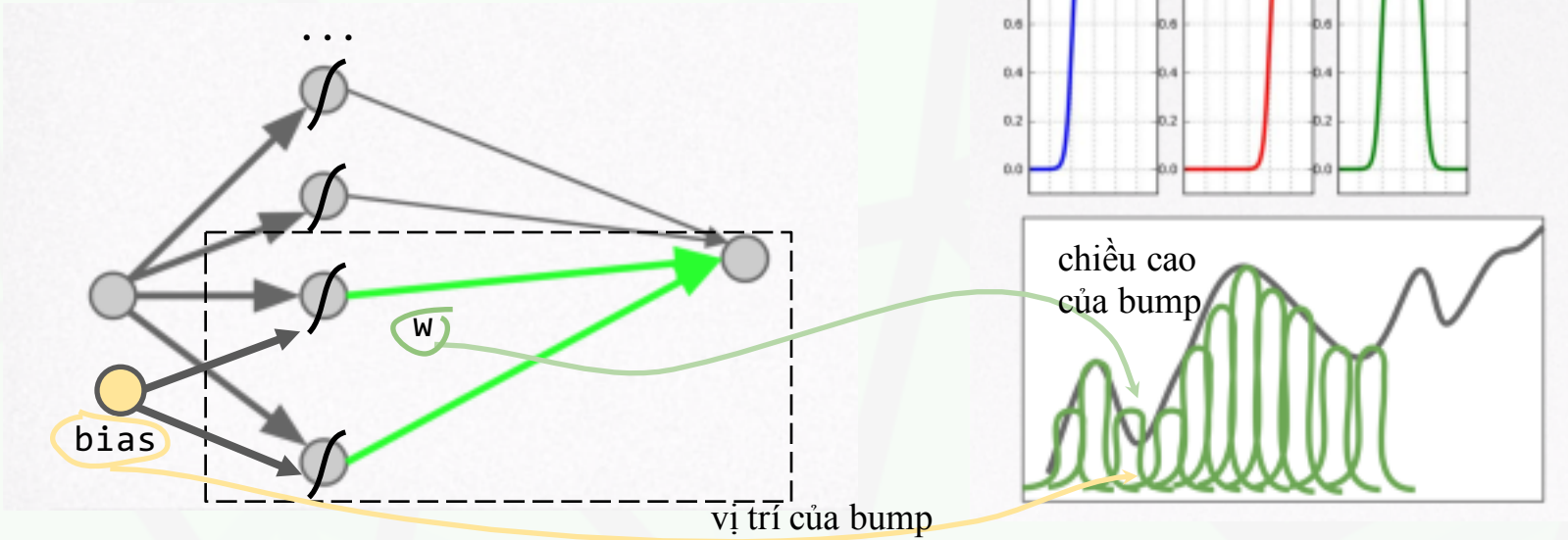




Neural Network

Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)

- 1 cặp hidden units (với hàm sigmoid) có thể tạo thành 1 "bump".
- Tổ hợp các bumps có thể tạo thành 1 hàm số bất kì

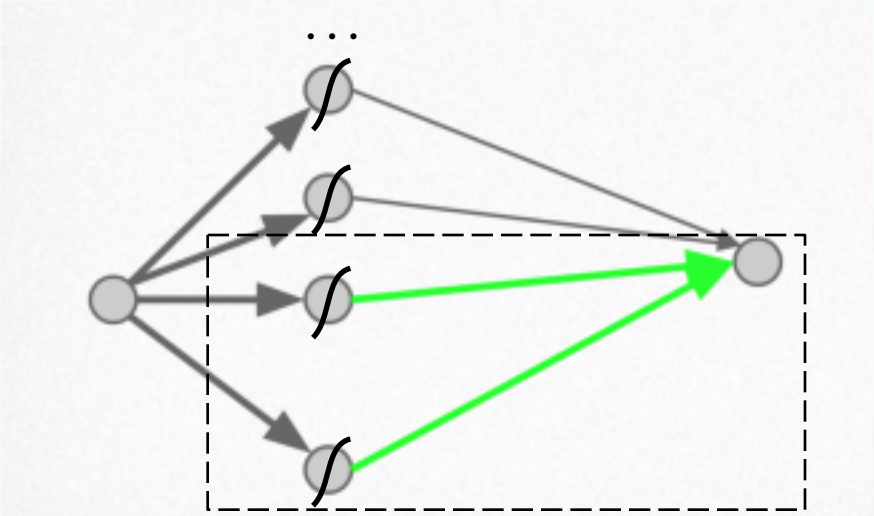


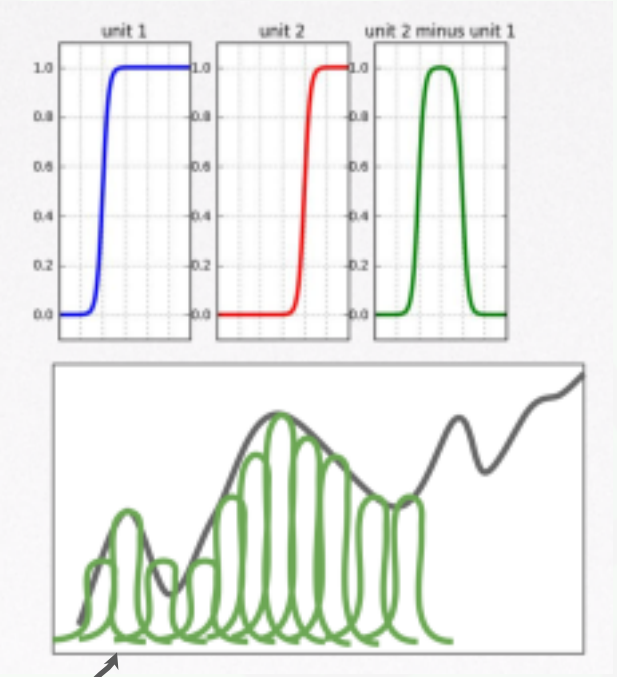
http://neuralnetworksanddeeplearning.com/movies/create_step_function.mp4

Neural Network

Một lớp ẩn được xem có thể xấp xỉ mọi hàm số (visual proof 1D)

- 1 cặp hidden units (với hàm sigmoid) có thể tạo thành 1 "bump".
- Tổ hợp **các bumps** có thể tạo thành 1 hàm số bất kì



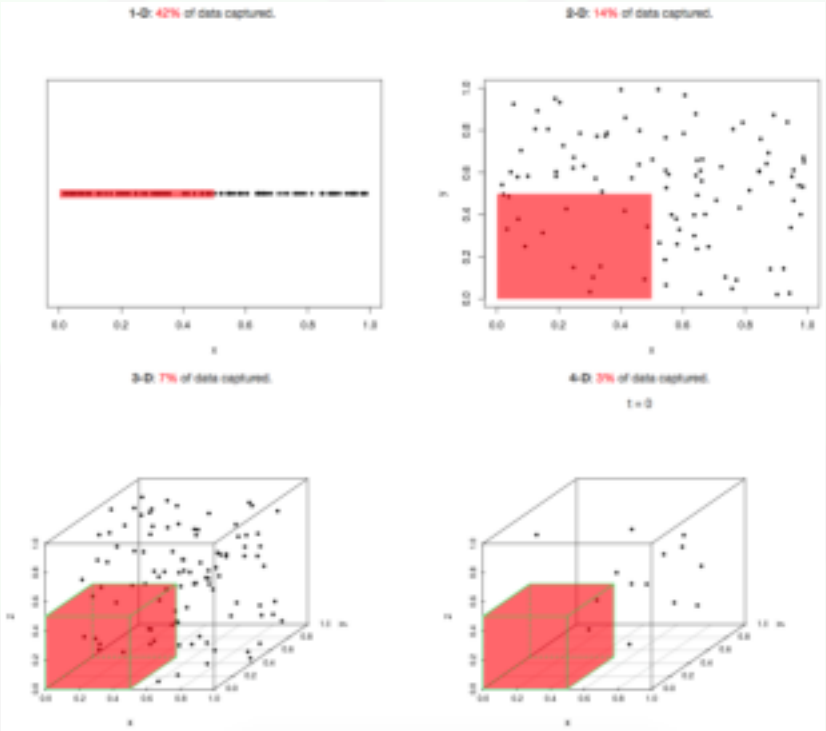


Dễ thấy sẽ cần **rất nhiều** cặp neurons như vậy

Neural Network

Tại sao cần mạng NN với nhiều hơn 1 hidden layer?

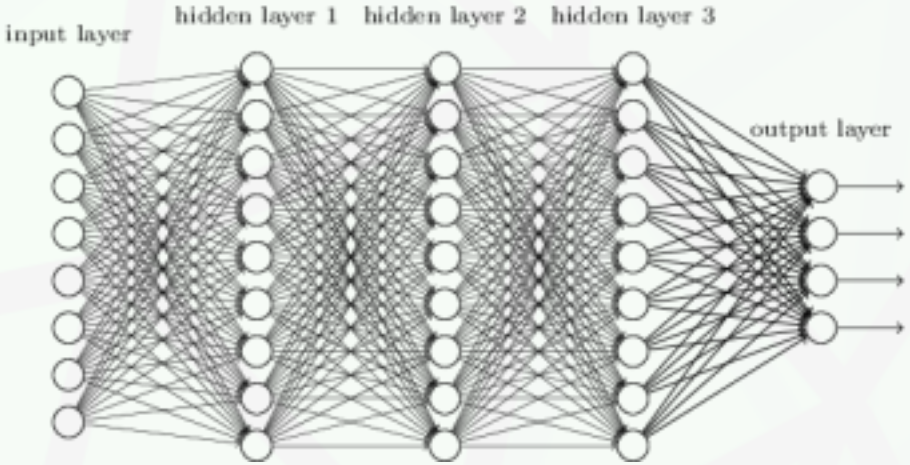
- 1 hidden layer cần rất nhiều hidden units để biểu diễn 1 arbitrary function
- Curse of dimensionality: Hiểu đơn giản, khi số chiều của dữ liệu tăng lên, *độ phức tạp* của model phải tăng lên theo **cấp số mũ** để đảm bảo khả năng giải quyết bài toán được giữ nguyên.



Neural Network

Tại sao cần mạng NN với nhiều hơn 1 hidden layer?

- 1 hidden layer cần rất nhiều hidden units để biểu diễn 1 arbitrary function
- Curse of dimensionality
- Có thể xem mỗi lớp như hàm mapping độc lập. Những lớp sau có thể tận dụng lại những hàm mapping đơn giản từ lớp trước đó để tạo ra hàm mapping có khả năng biểu diễn cực kì phức tạp (tính kế thừa) -> **cấu trúc phân cấp (III)**



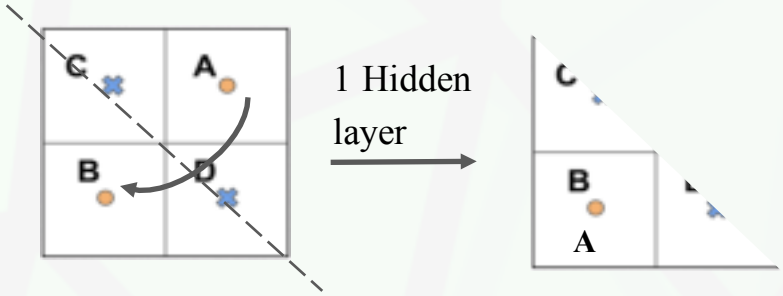
Ex: phép đếm -> phép cộng (đếm nhiều lần) -> phép nhân (cộng nhiều lần) -> phép mũ luỹ thừa (nhân nhiều lần)

Neural Network

Cấu trúc phân cấp (visual intuition)

Gấp không gian (Folding space)

- Một lớp hidden layer sẽ thực hiện phép biến đổi không gian phi tuyến, giúp quá trình phân loại ở các bước sau được dễ dàng hơn.
- Một trong những phép biến đổi phi tuyến khả dĩ là **gấp** những vùng không gian thuộc chung 1 lớp lại với nhau. (Xem lại ví dụ NN vs XOR problem)

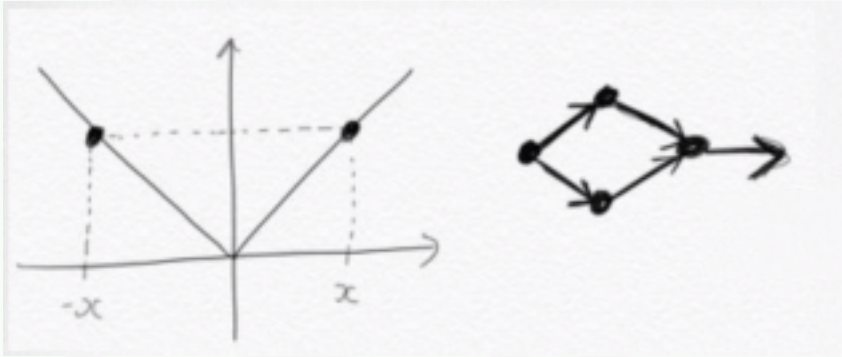


Neural Network

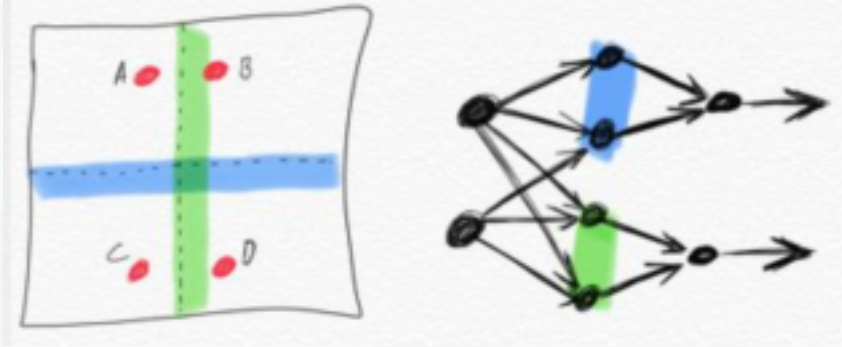
Cấu trúc phân cấp (visual intuition)

- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần. Chẳng hạn `f(x) = f(-x)` (Gấp đôi không gian ở x = 0)

Ex: $h_1 = Relu(-x)$, $h_2 = Relu(x)$
$f(x) = h_1 + h_2 = |x|$



1D



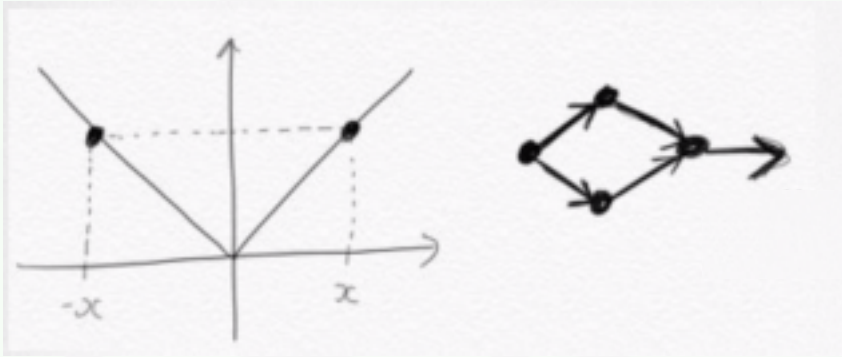
2D

Neural Network

Cấu trúc phân cấp (visual intuition)

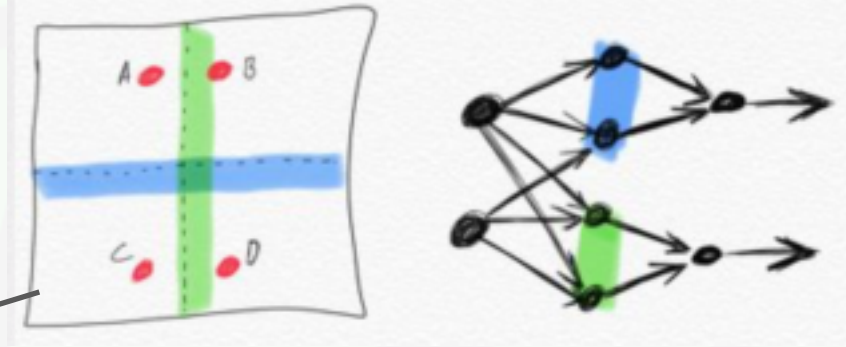
- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần. Chẳng hạn `f(x) = f(-x)` (Gấp đôi không gian ở x = 0)

Ex: $h_1 = Relu(-x), h_2 = Relu(x)$
$f(x) = h_1 + h_2 = |x|$



1D

Bài tập

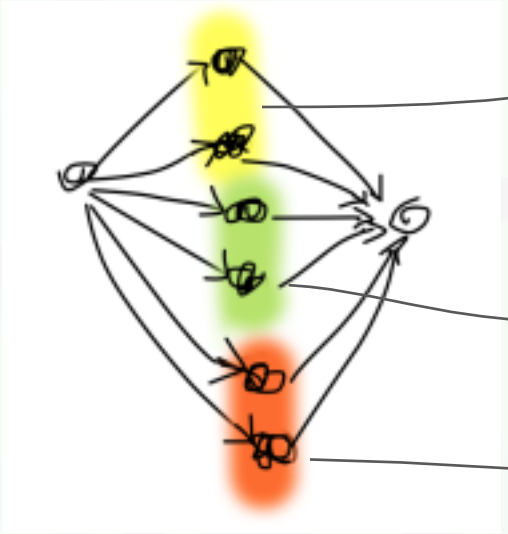


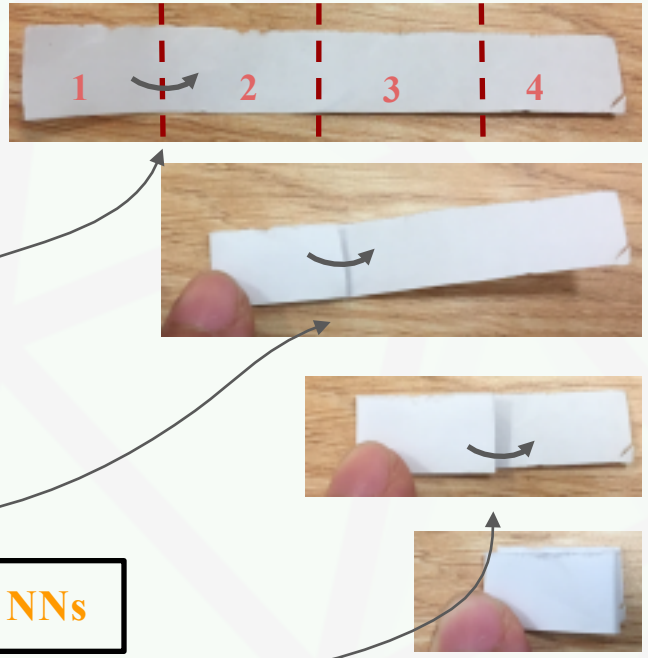
2D

Neural Network

Cấu trúc phân cấp (visual intuition)

- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần.



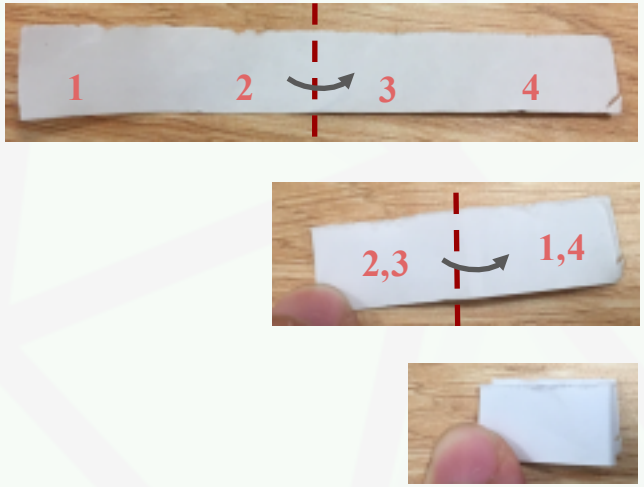


3 cặp NNs

Neural Network

Cấu trúc phân cấp (visual intuition)

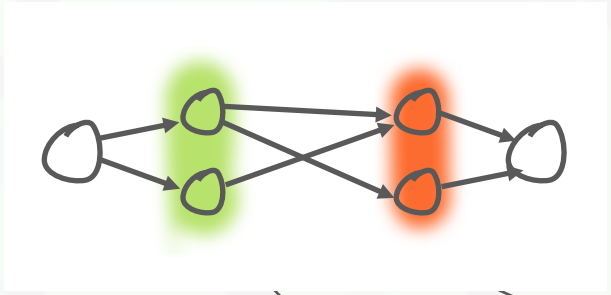
- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần.
- Nhiều vùng cùng chia sẻ chung 1 phép mapping. Ta có thể dùng ý tưởng này để giảm số phép toán.

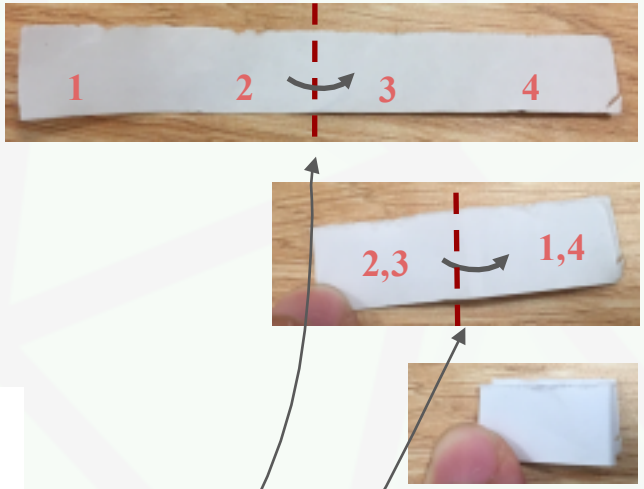


Neural Network

Cấu trúc phân cấp (visual intuition)

- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần.
- Nhiều vùng cùng chia sẻ chung 1 phép mapping. Ta có thể dùng ý tưởng này để giảm số phép toán.



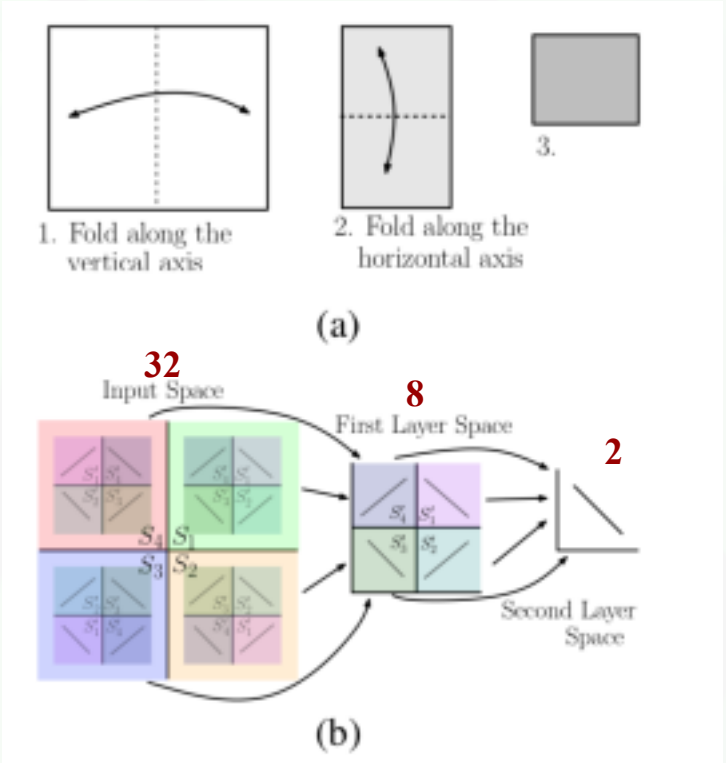


2 cặp NNs

Neural Network

Cấu trúc phân cấp (visual intuition)

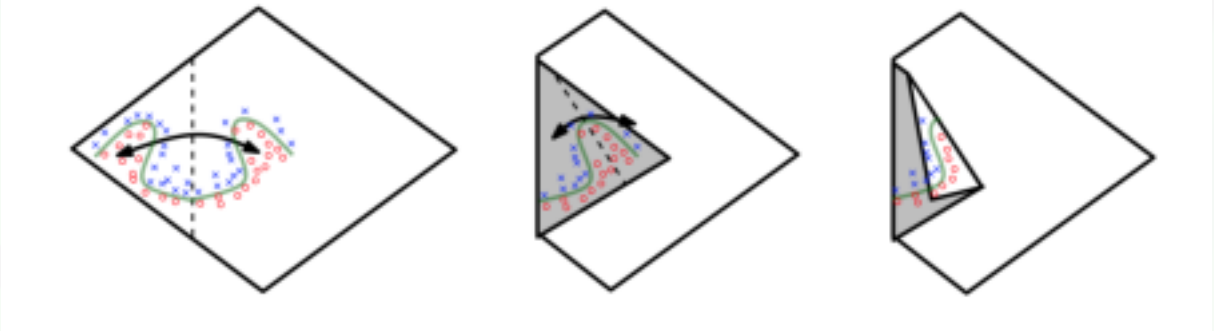
- Mỗi cặp NNs ở hidden layer với hàm ReLU có thể thực hiện phép biến đổi "gấp" không gian input của lớp hidden này 1 lần.
- Nhiều vùng cùng chia sẻ chung 1 phép mapping.
- Nếu thực hiện các hàm "folding space" này lần lượt (tương đương với qua nhiều lớp) sẽ dẫn tới việc phân chia số vùng ở cấp số mũ.



Neural Network

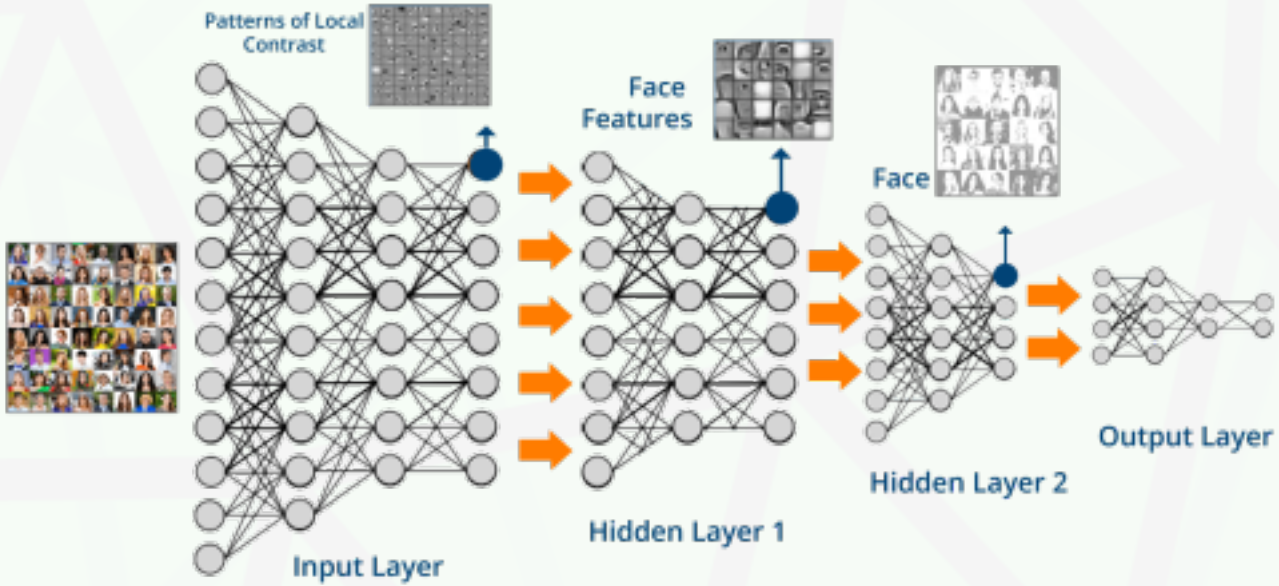
Cấu trúc phân cấp

* Trên thực tế, quá trình biến đổi không gian input ở các lớp hidden layer diễn ra phức tạp hơn và khó kiểm soát hơn so với những ví dụ trên.



Neural Network

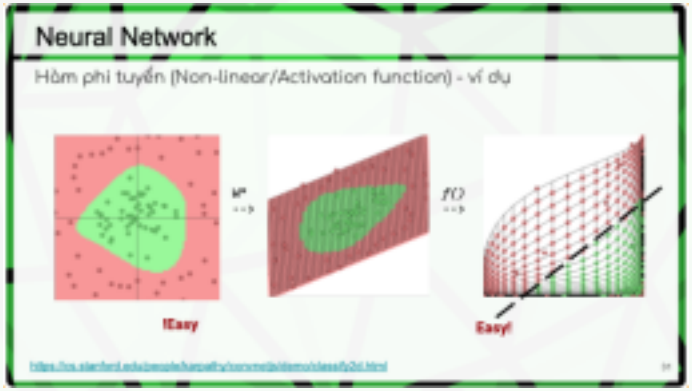
Cấu trúc phân cấp



- Cấu trúc phân cấp giúp phân đoạn hàm mapping phức tạp ra thành từng bước một với những hàm đơn giản hơn. Như ví dụ ở hình trên.

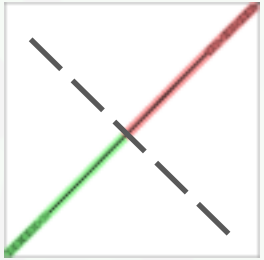
Neural Network

Cấu trúc phân cấp



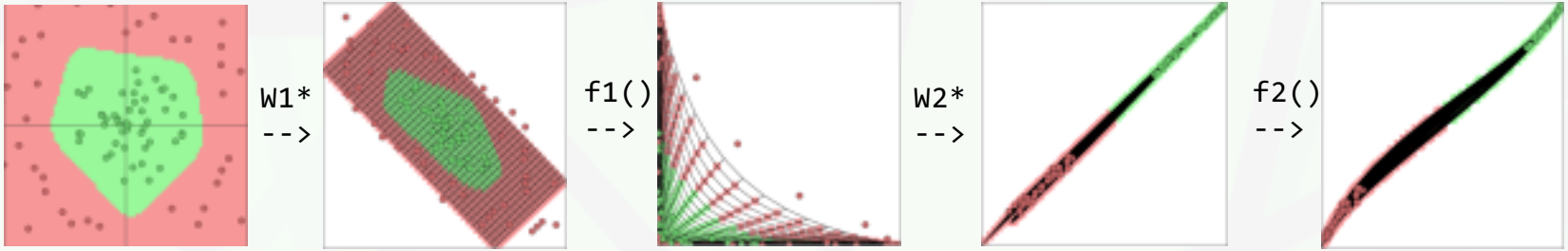
Quá trình biến đổi trở nên đơn giản hơn

NN với 1 lớp ẩn



W3*^

NN
với nhiều hơn 1 lớp ẩn



Deep Neural Network

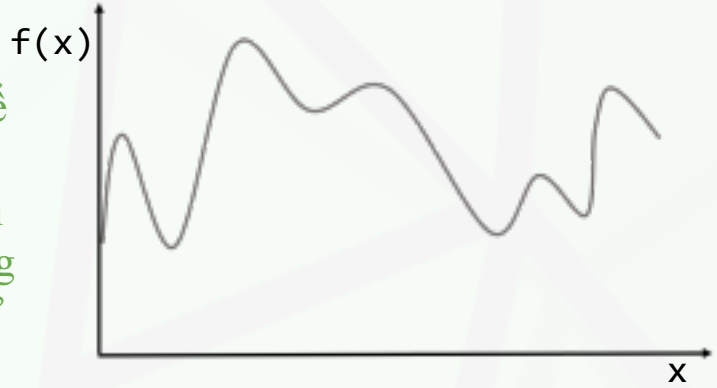
= Linear model (I)

+ Hàm phi tuyến (II)

+ Cấu trúc phân cấp(III)

Function Finder

Đi tìm hàm số

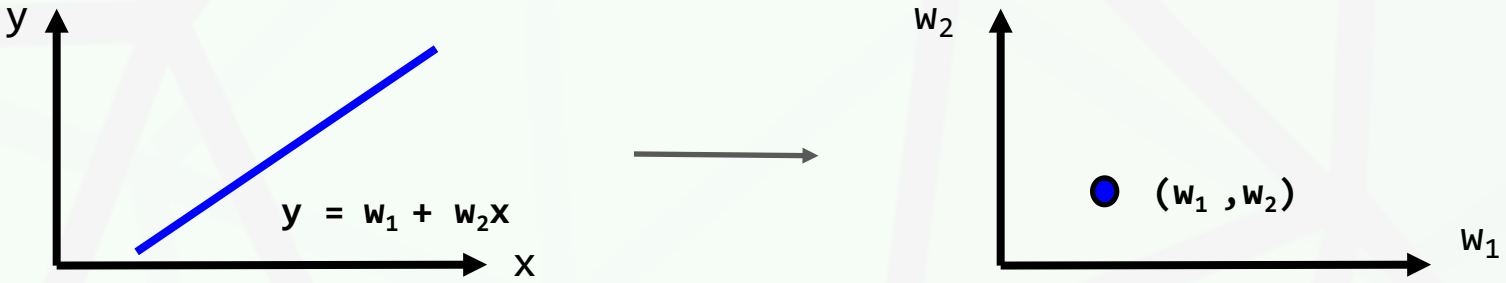


- Ta đã đề cập tới NNs như một "cỗ máy" có thể xấp xỉ bất kì hàm số nào.
- Tuy nhiên Universial Approximation Theorem (UAT) chỉ đề cập tới khả năng của NNs, không phải là một lời chỉ dẫn cho việc tìm ra "set-up" của cỗ máy này (set-up = tham số)

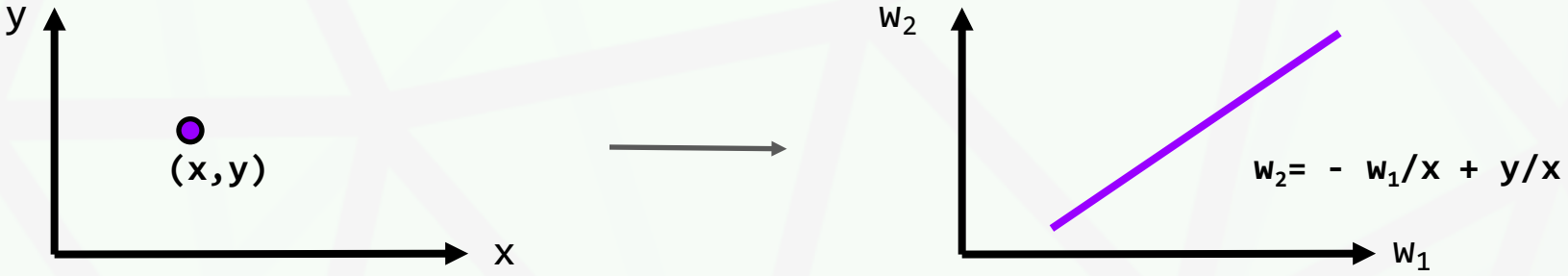
- **Làm sao để tìm ra các tham số của NNs để đạt được phép mapping như ý?**

Function Finder

Hough Transfrom: Biến đổi data space -> weight space



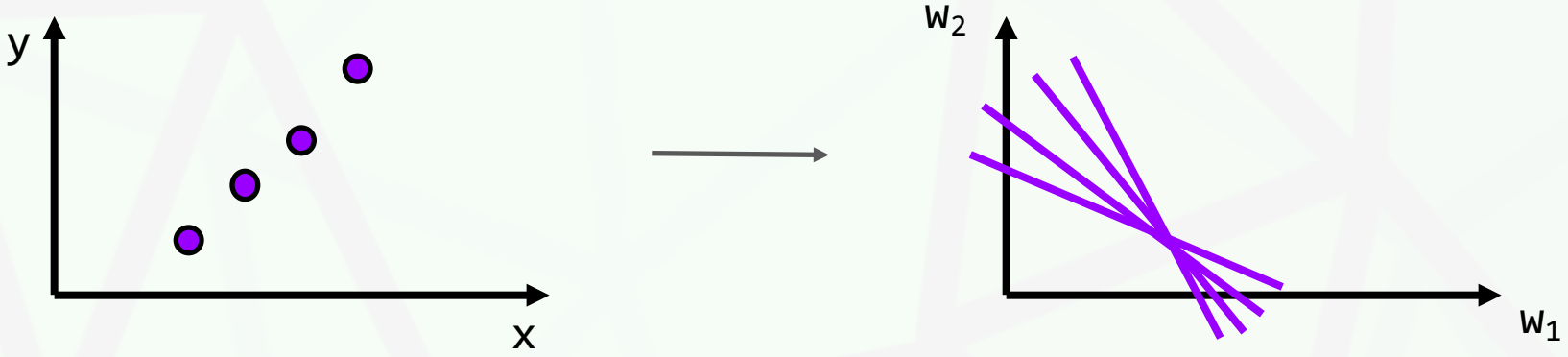
Mỗi đường thẳng trong không gian data tương đương với 1 điểm trong không gian trọng số



Mỗi điểm data có vô số đường thẳng đi qua, tập hợp trọng số của những đường thẳng này tạo thành một đường thẳng trong không gian trọng số.

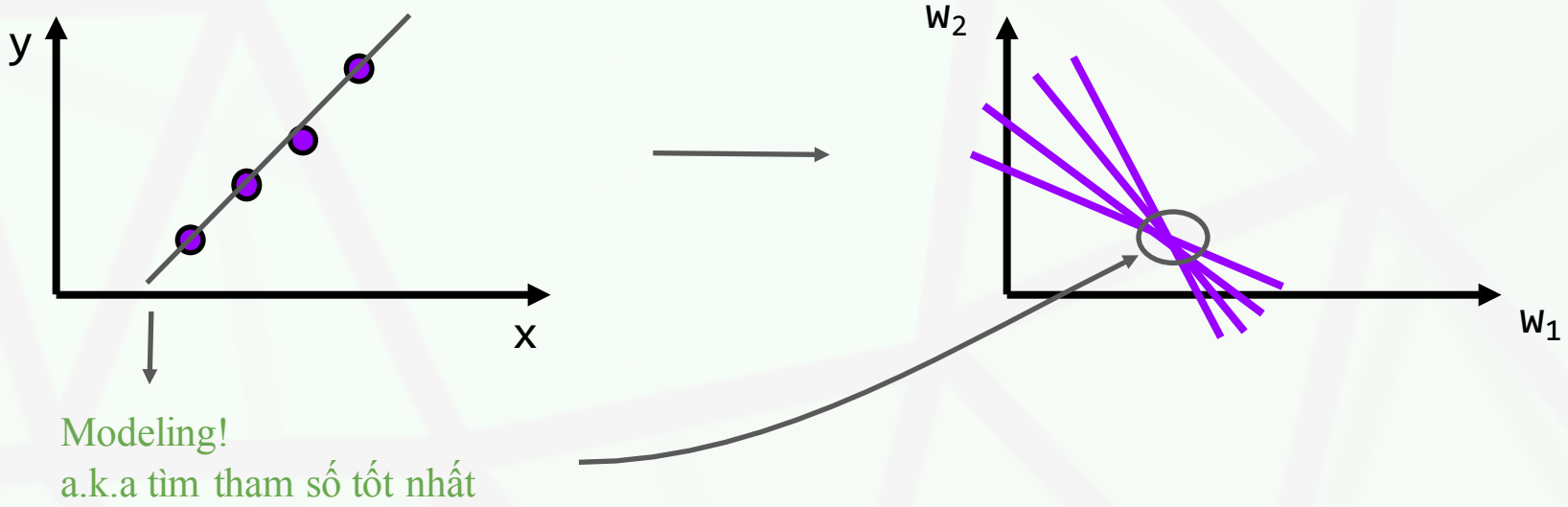
Function Finder

Hough Transfrom: Biến đổi data space -> weight space



Function Finder

Hough Transfrom: Biến đổi data space -> weight space

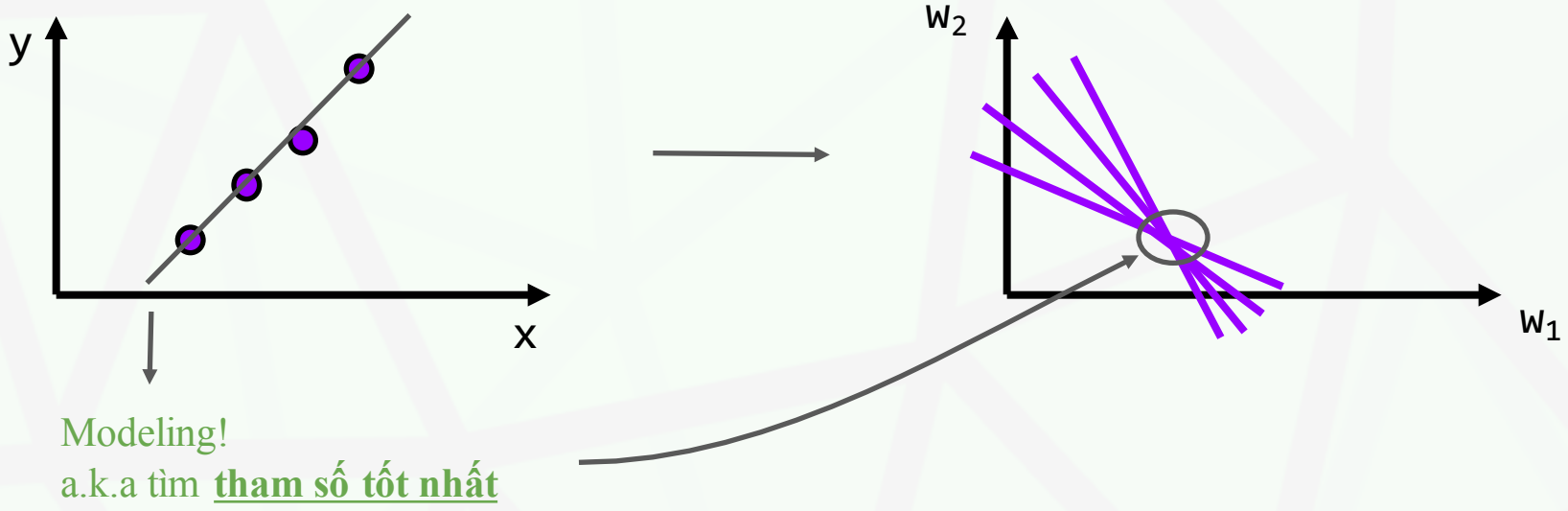


Modeling!
a.k.a tìm tham số tốt nhất

Có thể hình dung mỗi đáp án cho một bài toán giờ đây trở thành một điểm trong không gian tham số (giả định mọi bài toán đều có đáp án trong không gian này)

Function Finder

Hough Transfrom: Biến đổi data space -> weight space

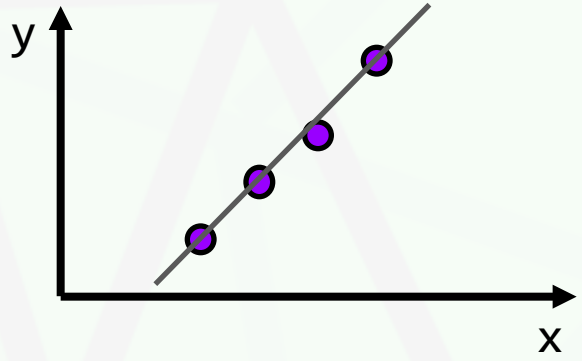


Modeling!
a.k.a tìm **tham số tốt nhất**

Để biết chính xác là đáp án tốt nhất với tiêu chí gì, ta đạt ra một hàm mục tiêu.
Hàm mục tiêu thường thấy: Mean square error (MSE)

Function Finder

Tìm hàm số (tham số)

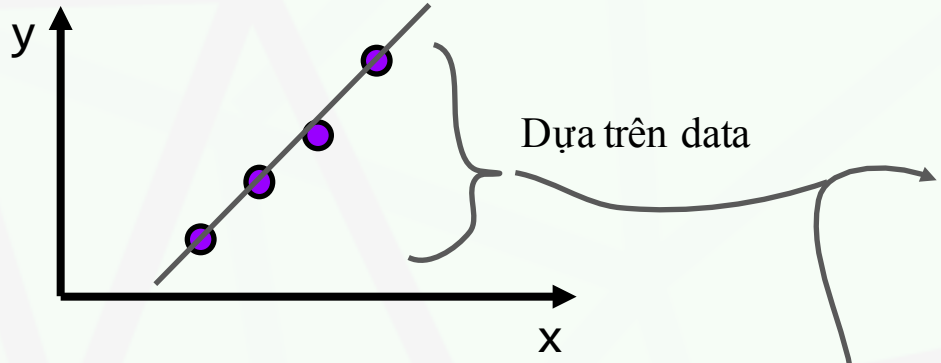


MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

Function Finder

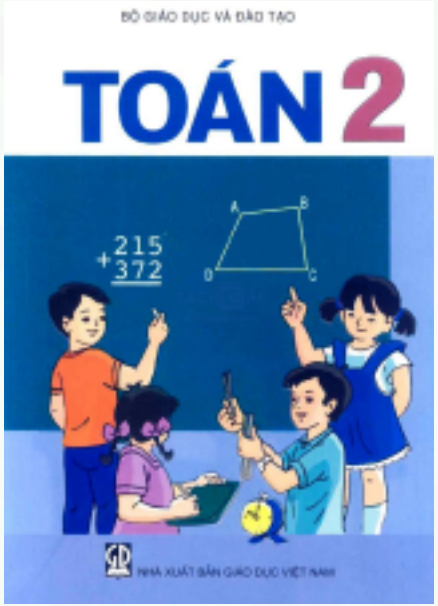
Tìm hàm số (tham số)



MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

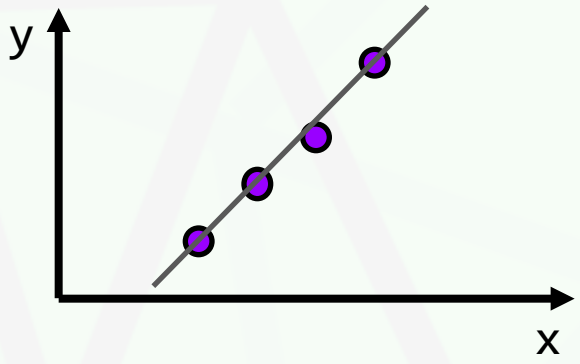
Analytical solution

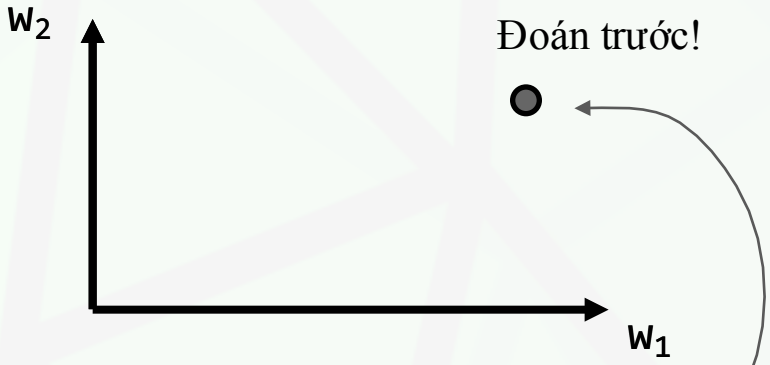


Your work!

Function Finder

Tìm hàm số (tham số)





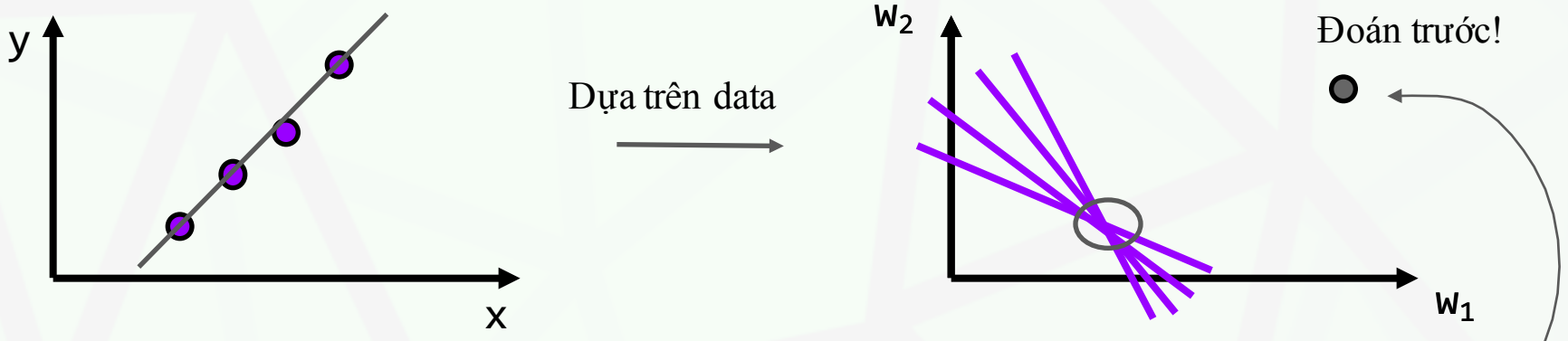
MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

Numerical solution: Gradient Decent

Function Finder

Tìm hàm số (tham số)



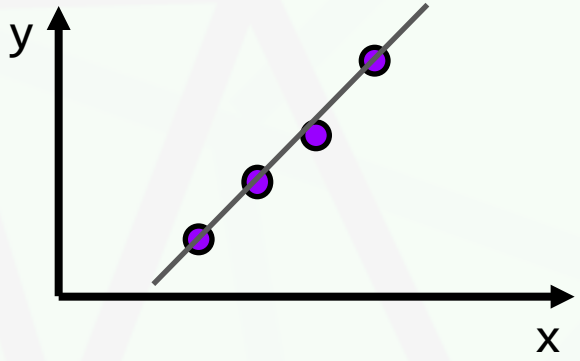
MSE: Loss function $L(f^*) = \Sigma_i \, ||y_i - f^*(x_i)||^2$

Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

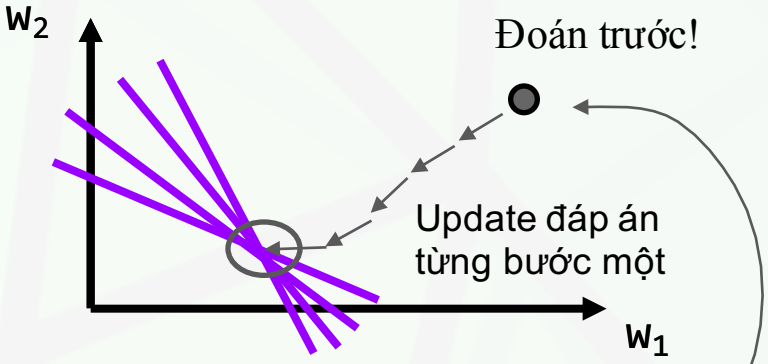
Numerical solution: Gradient Decent

Function Finder

Tìm hàm số (tham số)



Dựa trên data



MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

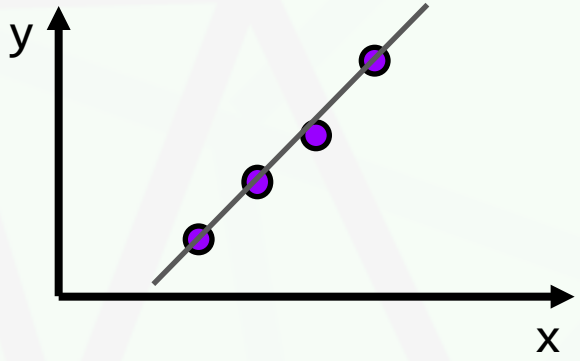
Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

Numerical solution: Gradient Decent

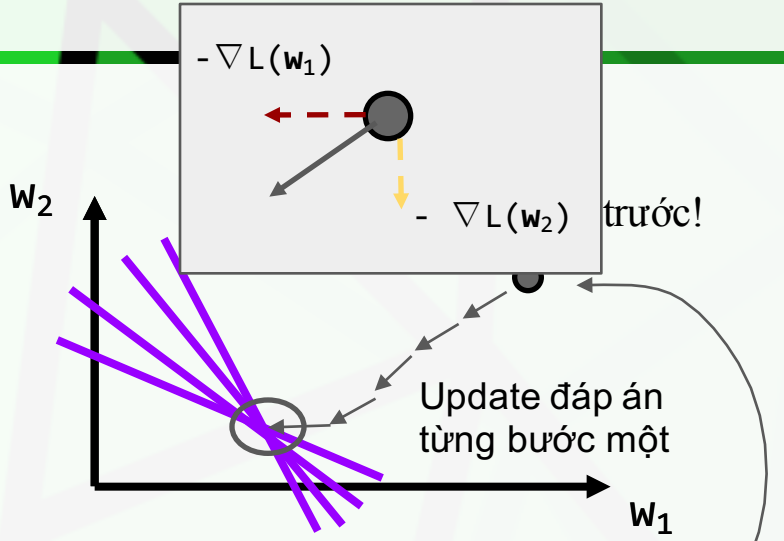
Sync all j: $w_{j(t+1)} = w_{j(t)} - \eta \Sigma_i \nabla L_i(w_j)$, for all i in |data|

Function Finder

Tìm hàm số (tham số)



Dựa trên data



MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

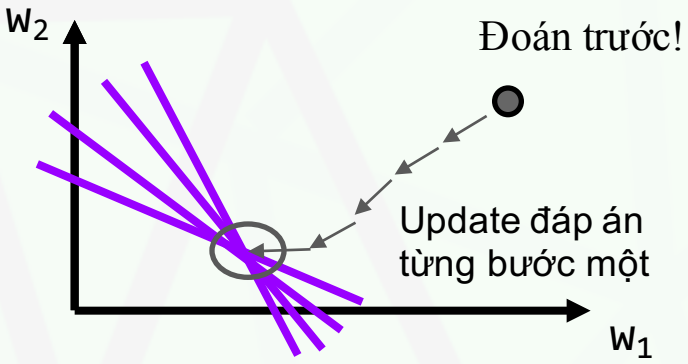
Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

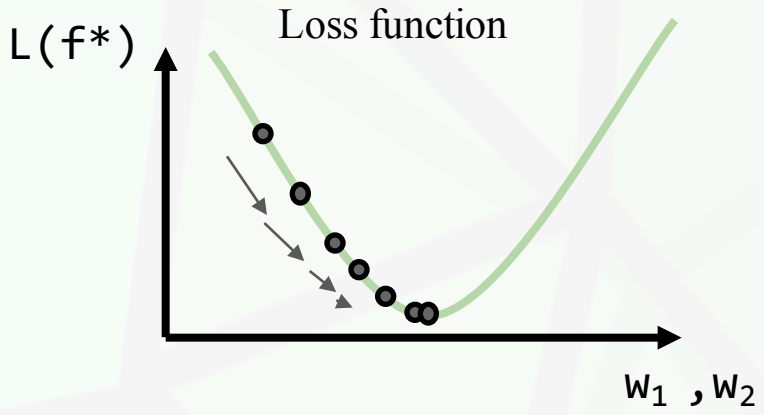
Numerical solution: Gradient Decent

Sync all j: $w_{j(t+1)} = w_{j(t)} - \eta \Sigma_i \nabla L_i(w_j)$, for all i in |data|

Function Finder

Tìm hàm số (tham số)





MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

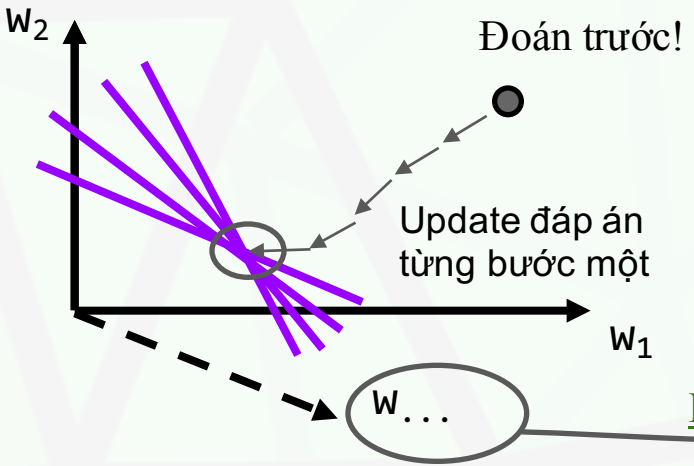
Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

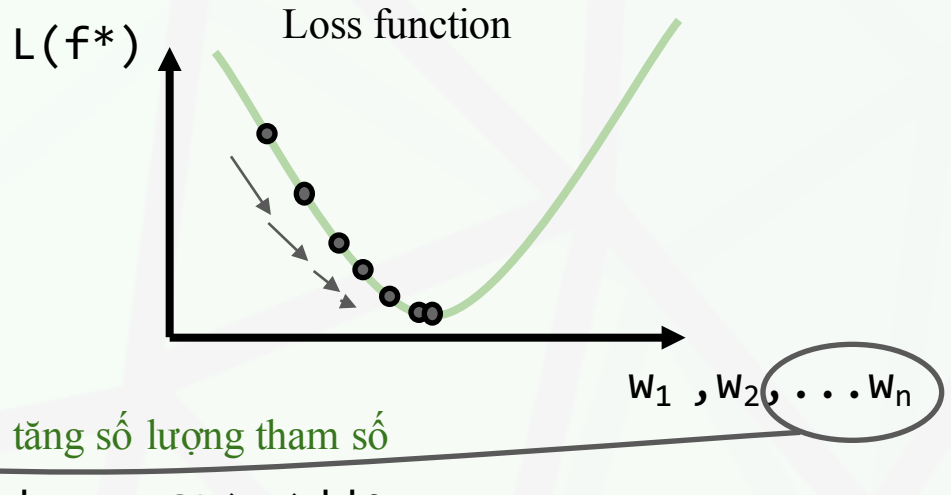
<u>Numerical solution:</u> Gradient Decent

Sync all j: $w_{j(t+1)} = w_{j(t)} - \eta \Sigma_i \nabla L_i(w_j)$, for all i in |data|

Function Finder

Tìm hàm số (tham số)





MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

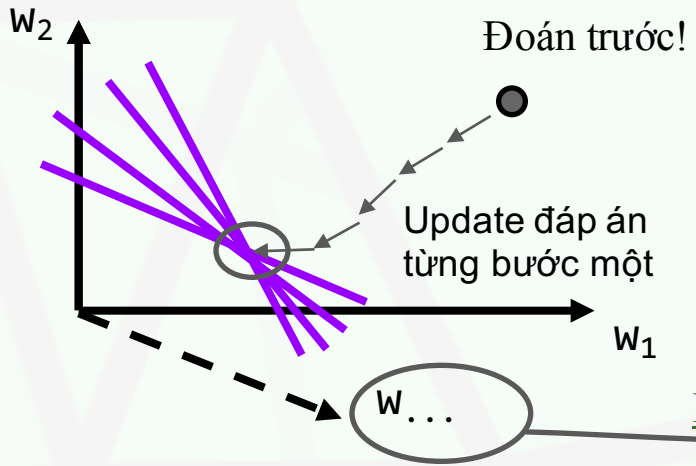
Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

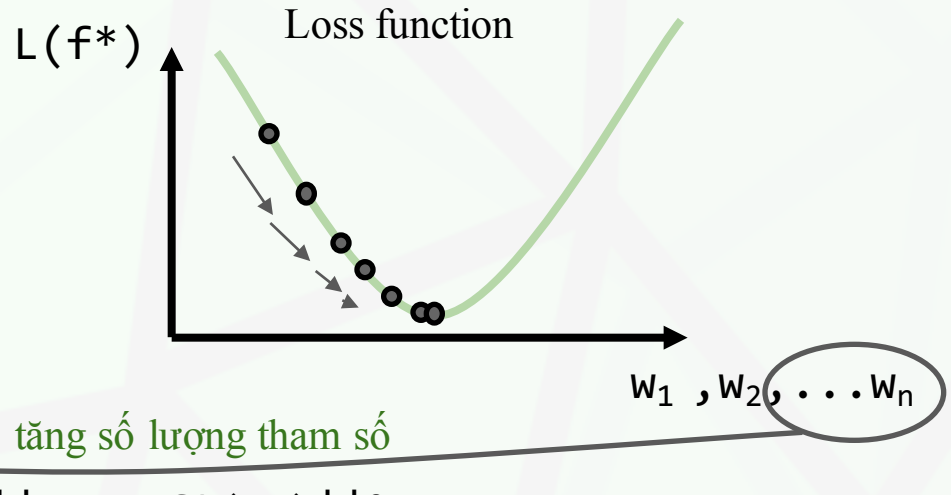
<u>Numerical solution:</u> Gradient Decent

Sync all j: $w_{j(t+1)} = w_{j(t)} - \eta \Sigma_i \nabla L_i(w_j)$, for all i in |data|

Function Finder

Tìm hàm số (tham số)





MSE: Loss function $L(f^*) = \Sigma_i ||y_i - f^*(x_i)||^2$

Tìm w_1, w_2: $\text{argmin}_{w1,w2}[L(f^*)]$

Không phải TH nào đạo hàm tham số cũng có thể tính đồng thời

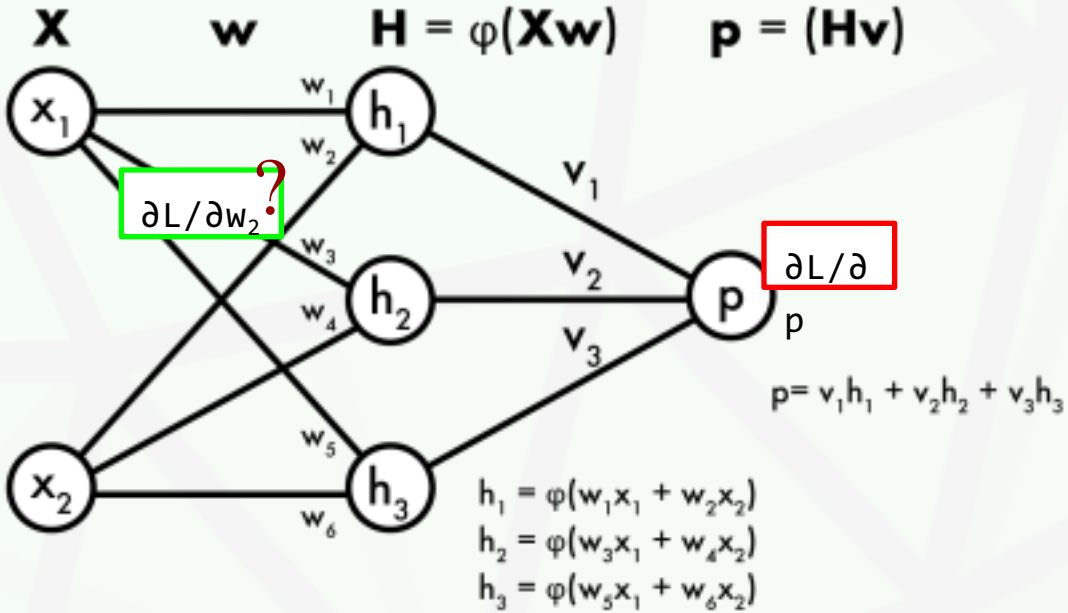
Numerical solution: Gradient Decent

Sync all j: $w_{j(t+1)} = w_{j(t)} - \eta \Sigma_i \nabla L_i(w_j)$, for all i in |data|

Hierarchical Function Finder

Tìm hàm số (tham số)

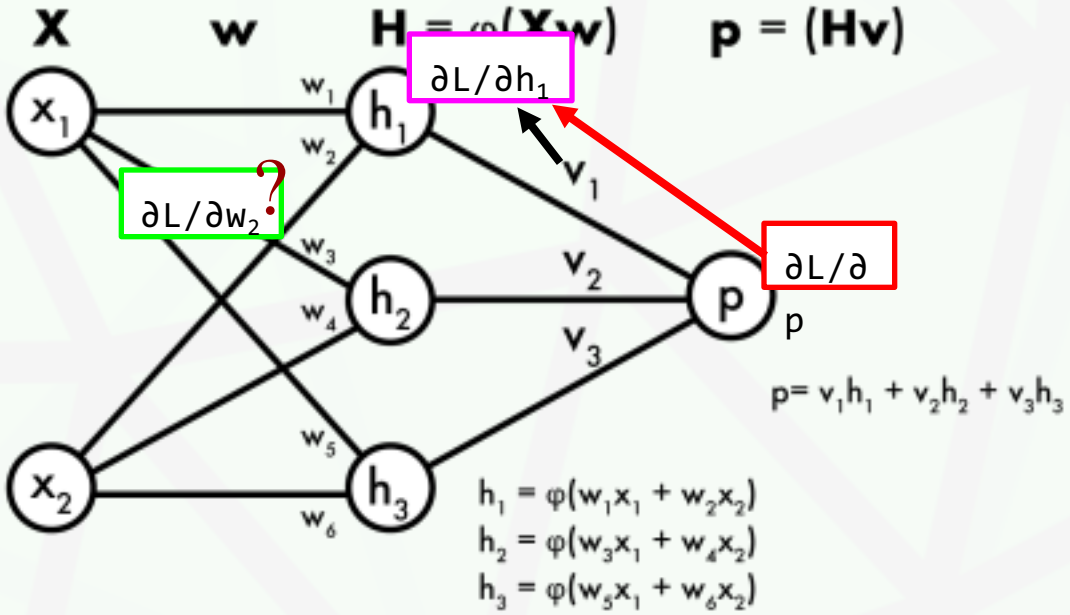
Back Propagation



Hierarchical Function Finder

Tìm hàm số (tham số)

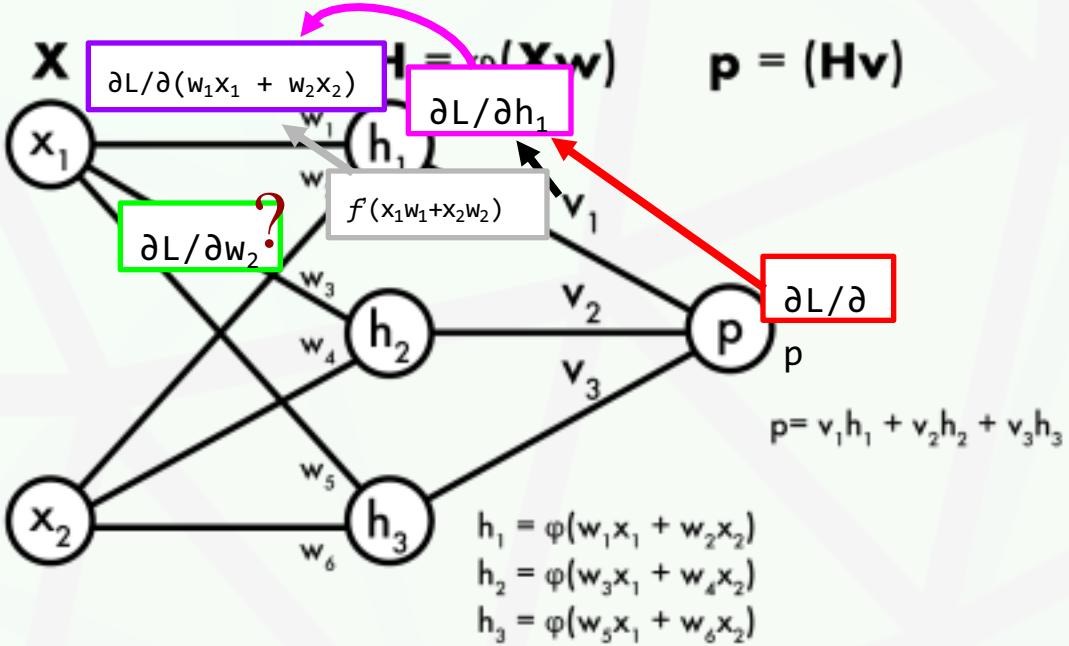
Back Propagation



Hierarchical Function Finder

Tìm hàm số (tham số)

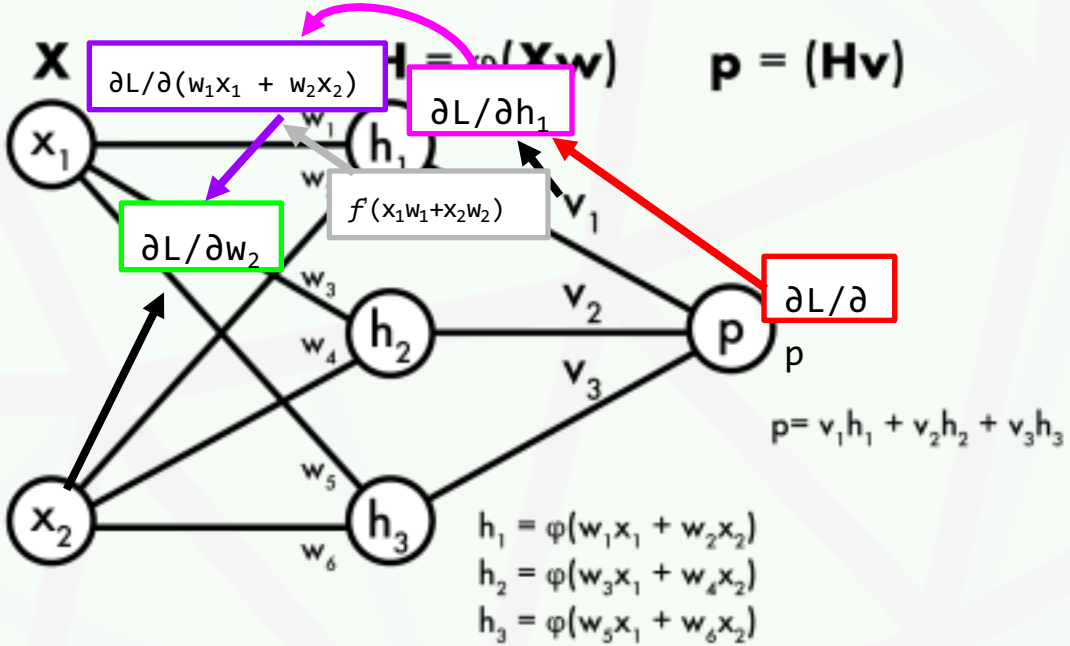
Back Propagation



Hierarchical Function Finder

Tìm hàm số (tham số)

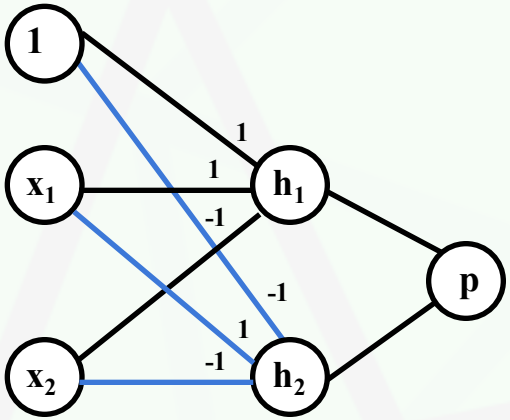
Back Propagation

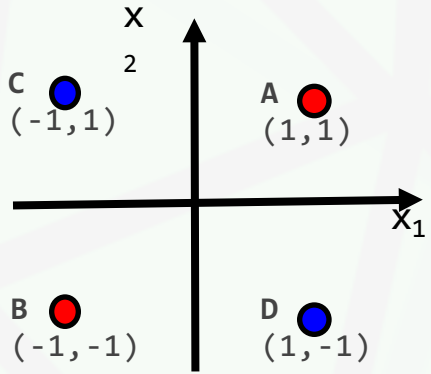


Sẽ quay lại trong phần Guest Lecture

Neural Network

Một lớp ẩn vs XOR problem





XOR

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html