TRÍ TUỆ NHÂN TẠO

Bài 15: Mạng thần kinh nhân tạo (2)

Nội dung

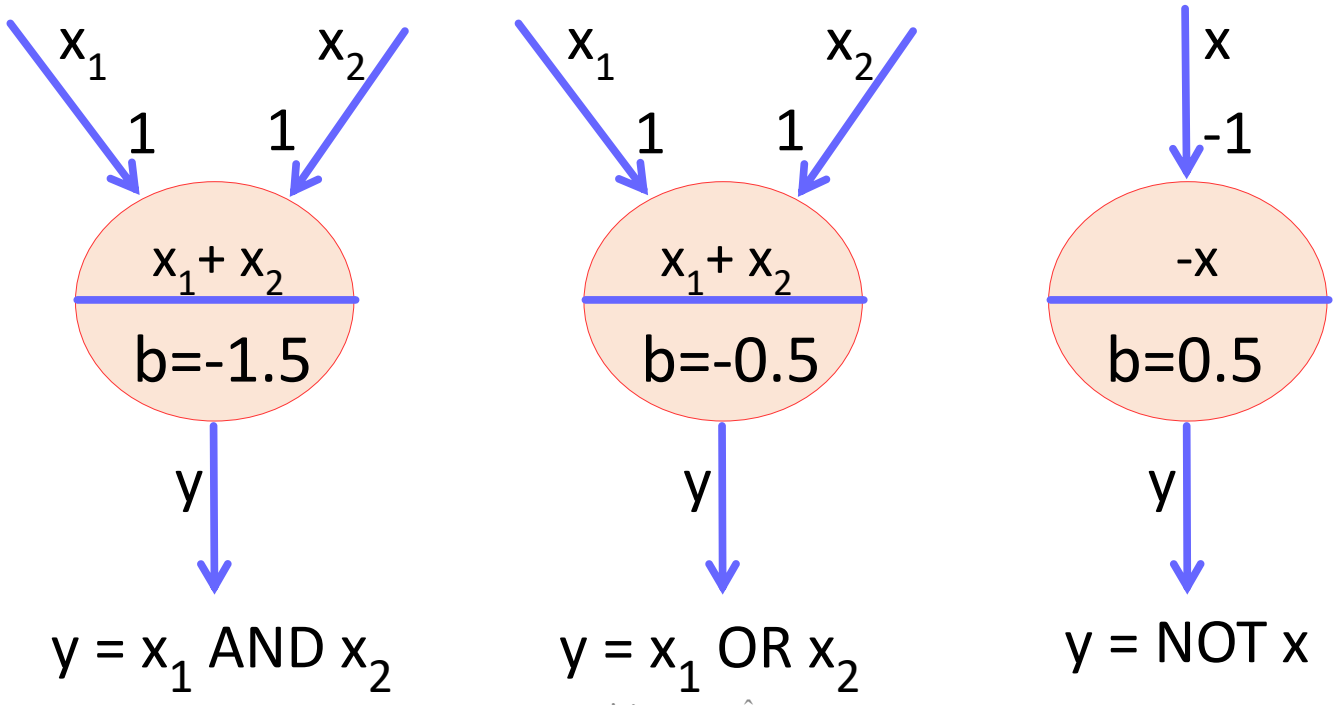
1. Mạng các perceptron
2. Học sâu (deep learning)
3. Mạng tích chập (CNN)
4. Bộ tự mã hóa (autoencoder)
5. Bắt đầu với học sâu như thế nào?

Phần 1

Mạng các perceptron

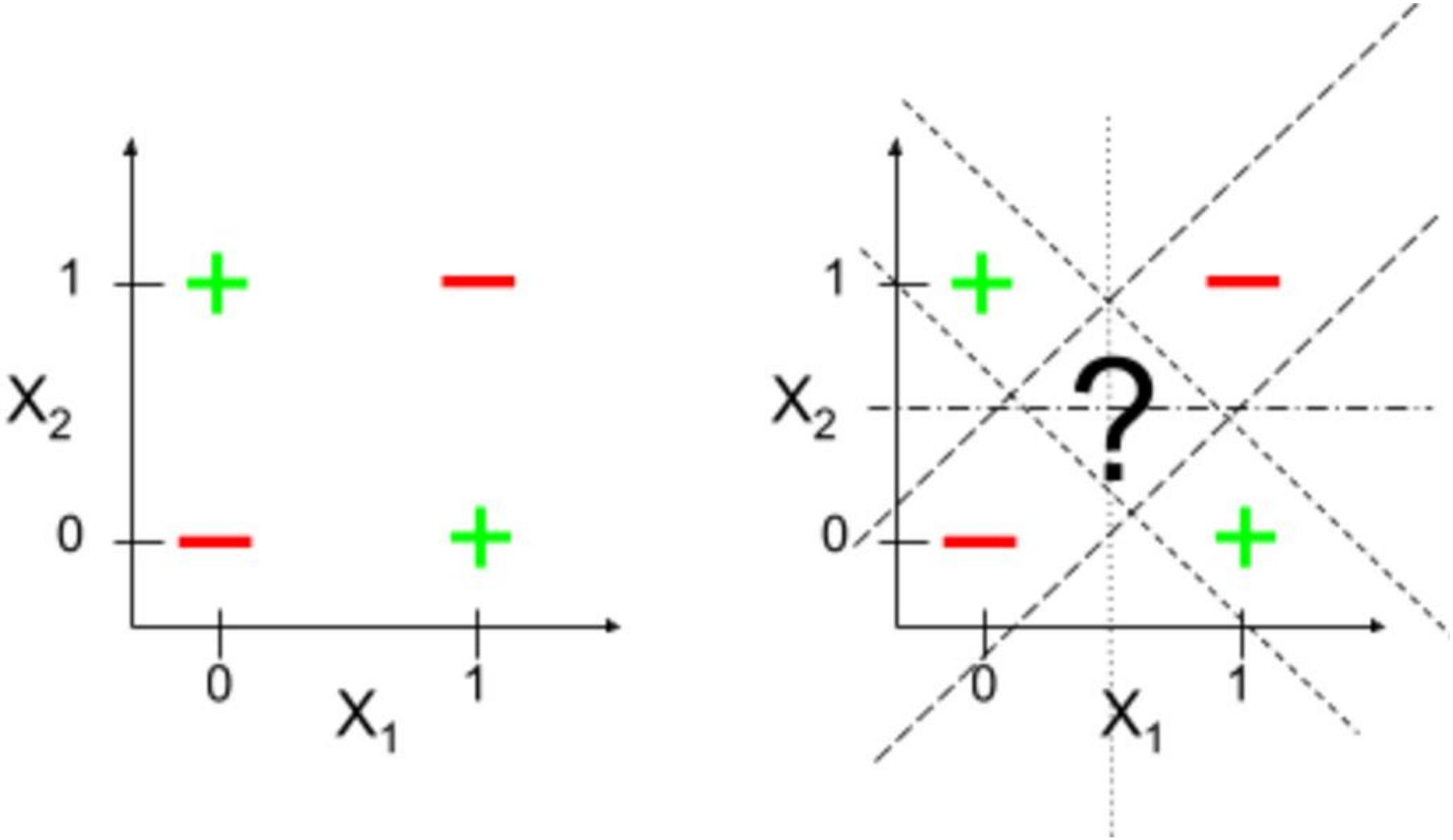
Sức mạnh của một perception

- Một perception mạnh hơn một cổng logic cơ bản
- Ví dụ một perception dùng hàm kích hoạt hardlim, 2 đầu vào, $w_1 = 1$, $w_2 = 1$
  - Chọn b = -1.5 ta được cổng AND
  - Chọn b = -0.5 ta được cổng OR



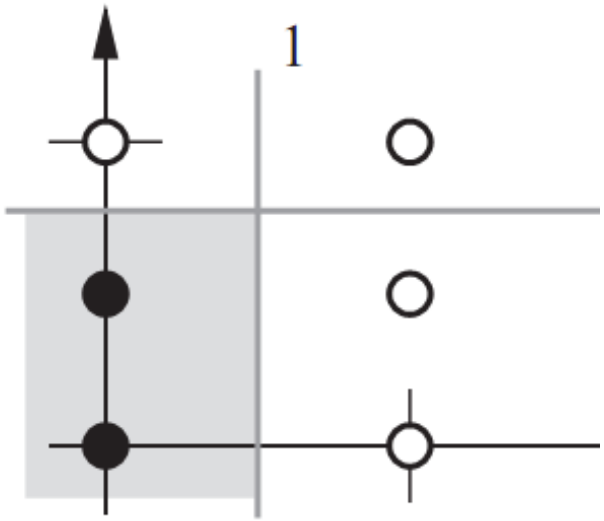
Hạn chế của một perceptron

- Không giải quyết được bài toán XOR



Mạng các perceptron

- Các perceptron có thể kết nối với nhau thành mạng lưới và mô phỏng các luật logic

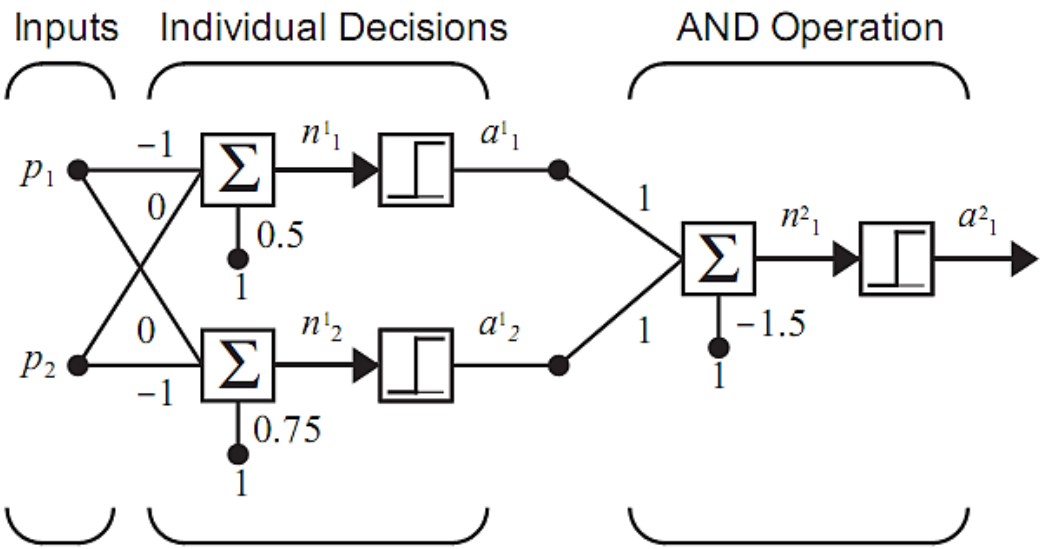


First Boundary:

$$a_1^1 = hardlim(\begin{bmatrix}-1 & 0\end{bmatrix}\mathbf{p} + 0.5)$$

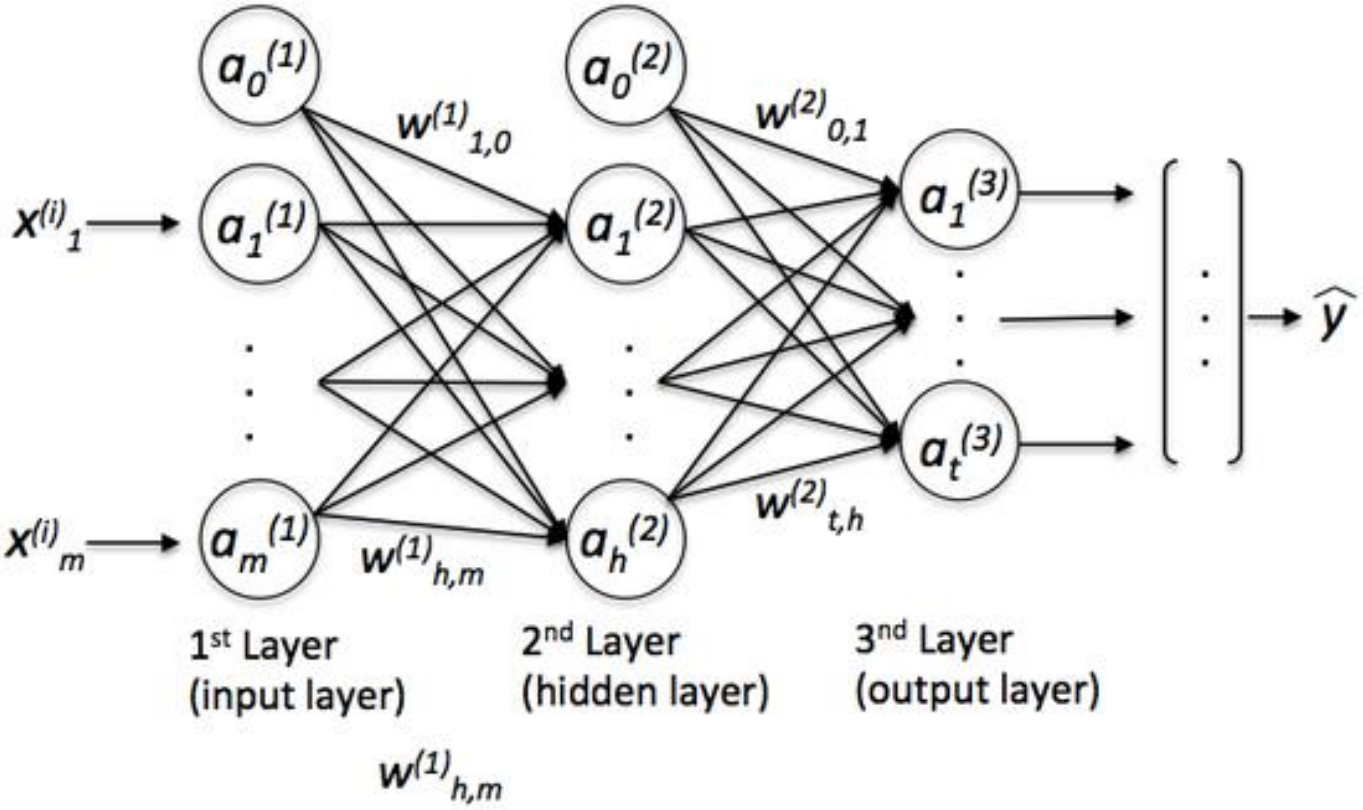
Second Boundary:

$$a_2^1 = hardlim(\begin{bmatrix}0 & -1\end{bmatrix}\mathbf{p} + 0.75)$$



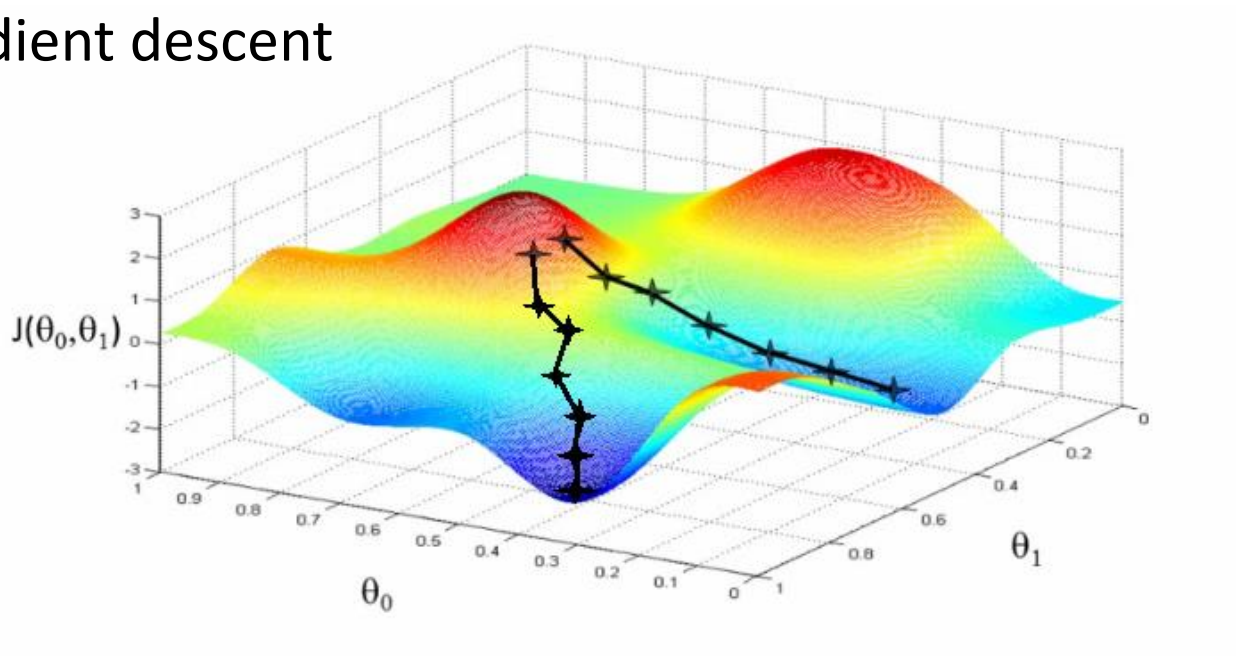
Mạng các perceptron

- Đặc tính xấp xỉ vạn năng: mạng neural nhân tạo 1 lớp ẩn có thể xấp xỉ hàm số liên tục bất kỳ
- Huấn luyện: lan truyền ngược lỗi



Bản chất việc huấn luyện

- Lỗi = hàm đánh giá
- Huấn luyện ~ cực tiểu hóa hàm số
- Ý tưởng: đi ngược hướng đạo hàm
  - Gradient descent
  - Stochastic gradient descent
  - Minibatch gradient descent



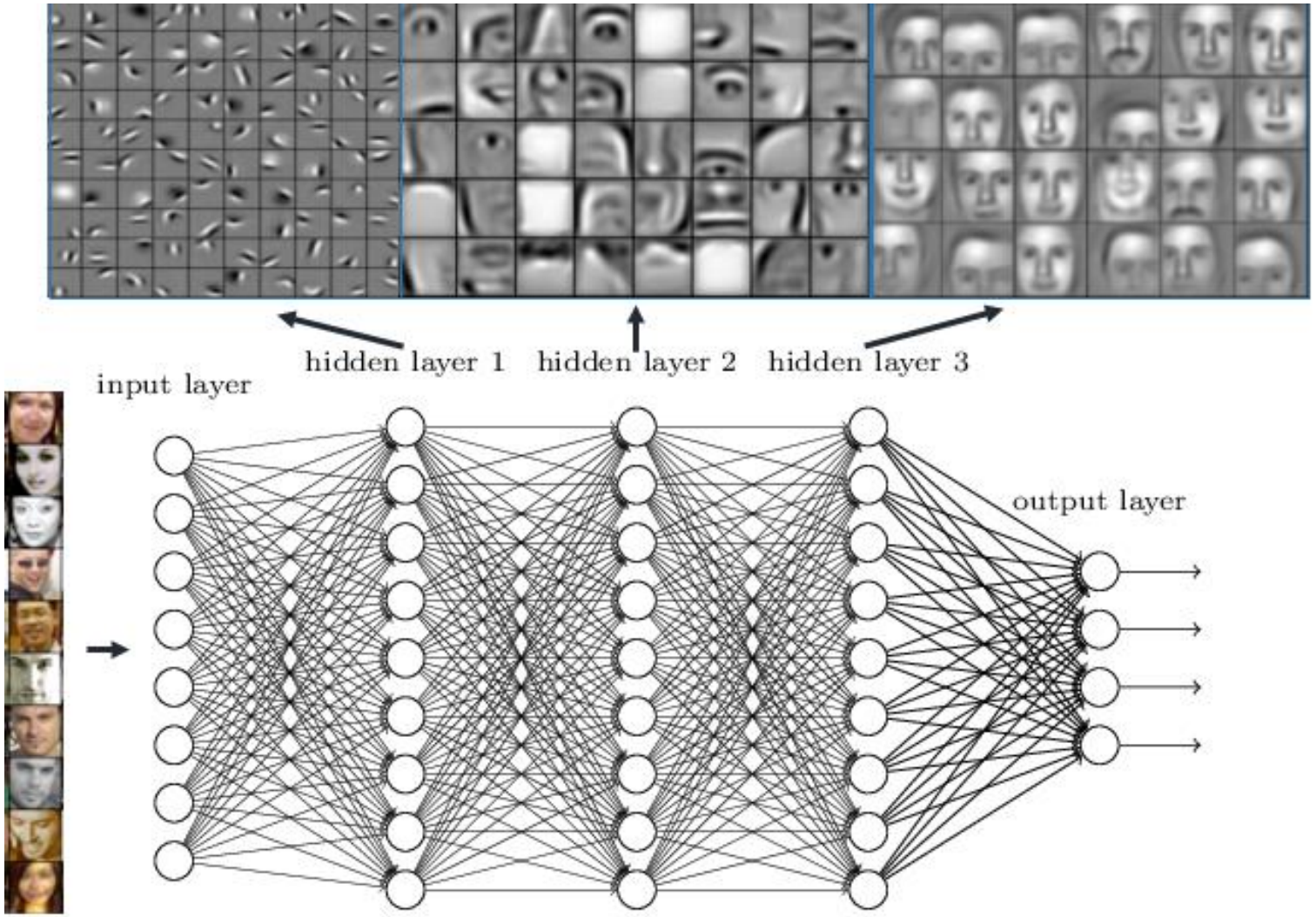
Phần 2

Học sâu

Học sâu

- Bản chất là ANN có nhiều lớp
- Tại sao ngày trước không nghĩ đến?
  - Quy mô mạng quá lớn: một ANN có 1000 đầu vào, 2 lớp ẩn 500 nút, 10 đầu ra sẽ có 2,5 tỉ tham số
  - Quy mô mạng lớn đòi hỏi công suất tính toán lớn
  - Sự suy giảm quá nhanh của gradient trong các thuật toán tập huấn
- Điều chỉnh:
  - Kiến trúc mạng
  - Thuật toán huấn luyện
  - Ý đồ thiết kế của từng lớp

Học sâu

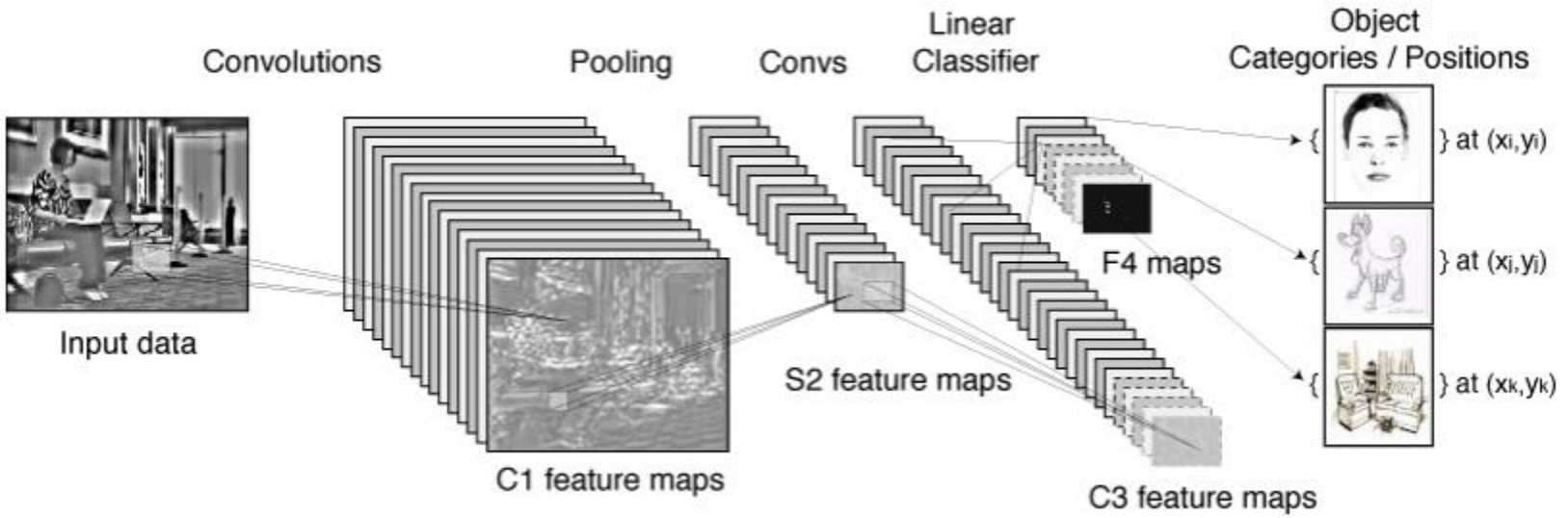

hidden layer 1
hidden layer 2
hidden layer 3
input layer
output layer

Phần 3

Mạng tích chập (CNN)

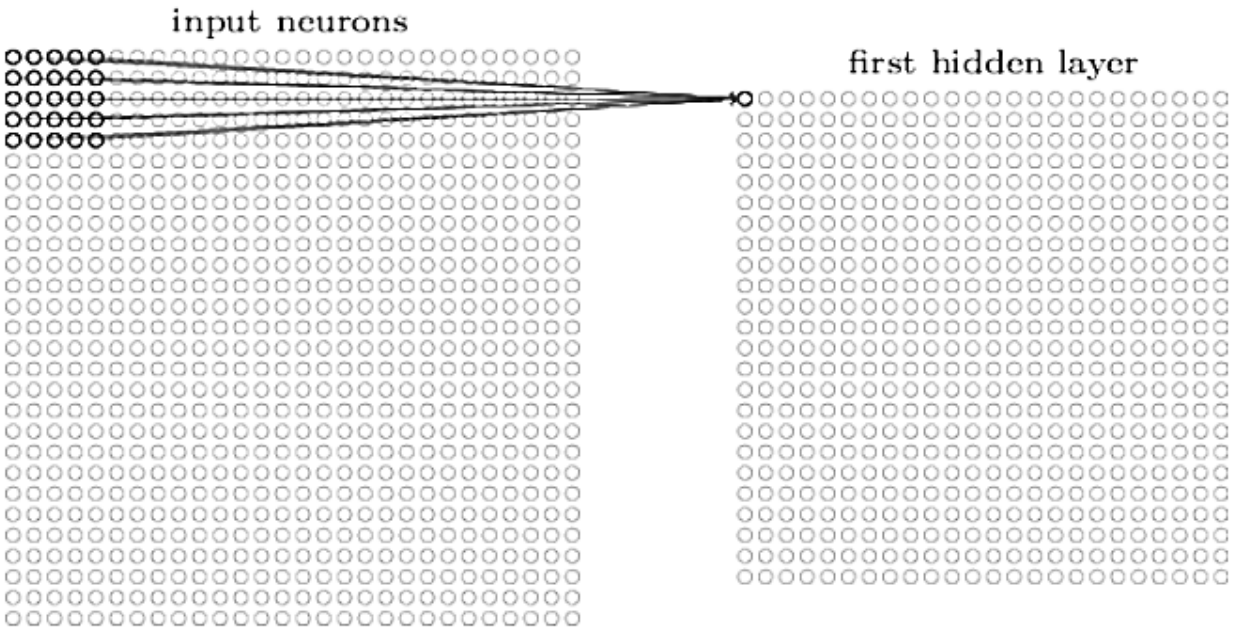
Convolutional Neural Networks

- CNN sử dụng 3 ý tưởng chính để điều chỉnh ANN:
  - LRF (local receptive fields)
  - Chia sẻ trọng số cùng lớp (shared weights)
  - Pooling



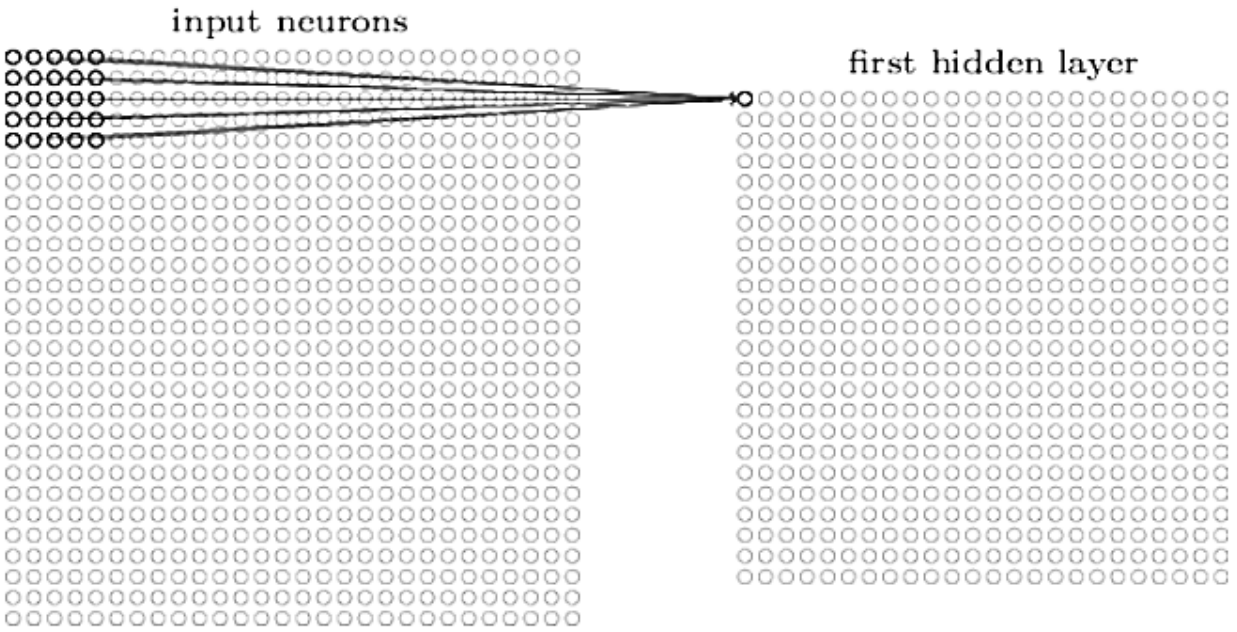
LRF (local receptive fields)

- Thay vì kết nối đầy đủ giữa 2 lớp mạng liên tiếp, một neural ở lớp sau chỉ kết nối với một vùng của lớp trước (gọi là vùng LRF của neural đó)
- Trượt LRF qua toàn bộ lớp trước tạo thành lớp sau



LRF (local receptive fields)

- Nếu ảnh cỡ 28x28 và vùng LRF cỡ 5x5 thì lớp phía sau sẽ là ma trận perception cỡ 24x24
- Mục đích của LRF: giảm số tham số của mạng hình thành tầng features cơ bản của ảnh



Chia sẻ trọng số cùng lớp

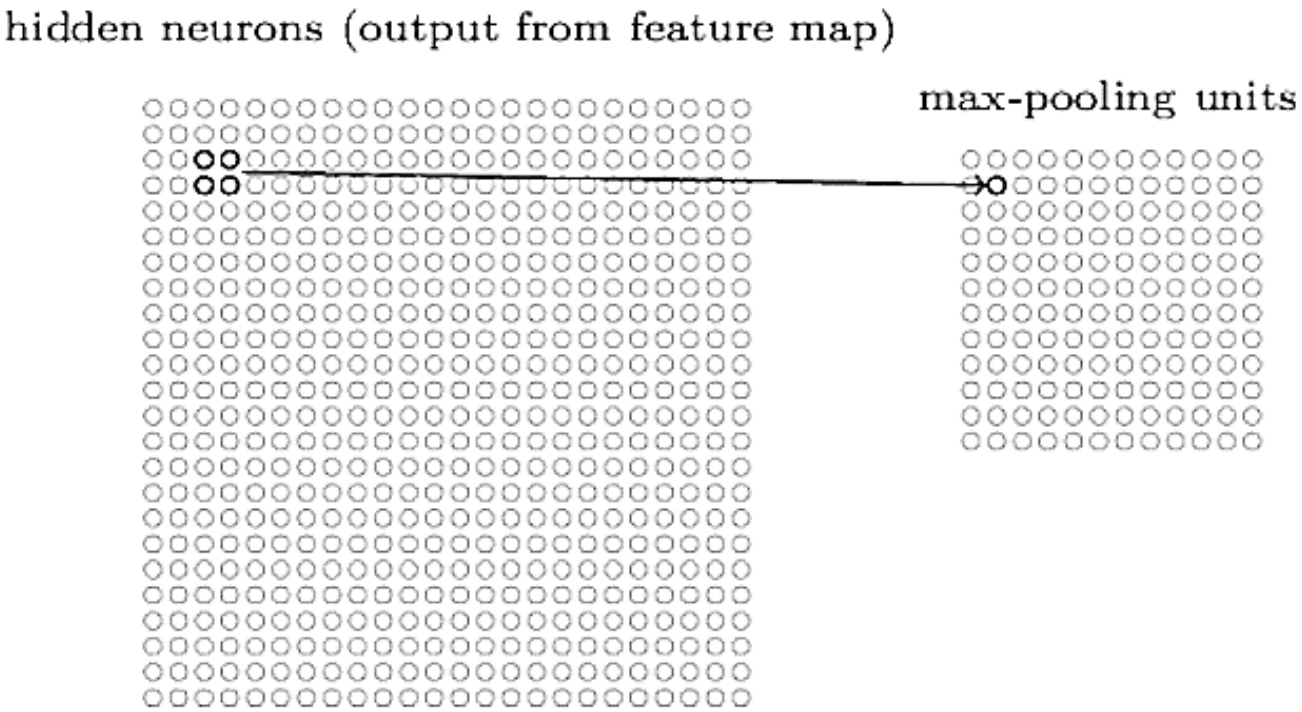
- Tất cả các neural thuộc cùng lớp sẽ dùng chung trọng số và bias
- Như trong ví dụ trước, tất cả 24x24 neural đều dùng chung ma trận trọng số W và hệ số b (bias)
- Như vậy output của neural (j,k) sẽ là

$$f\left(b+\sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

- Việc chia sẻ trọng số này đảm bảo tất cả các features của tầng này được xử lý như nhau
- Tăng tốc độ tính toán trong trường hợp sử dụng các phép toán ma trận của GPU

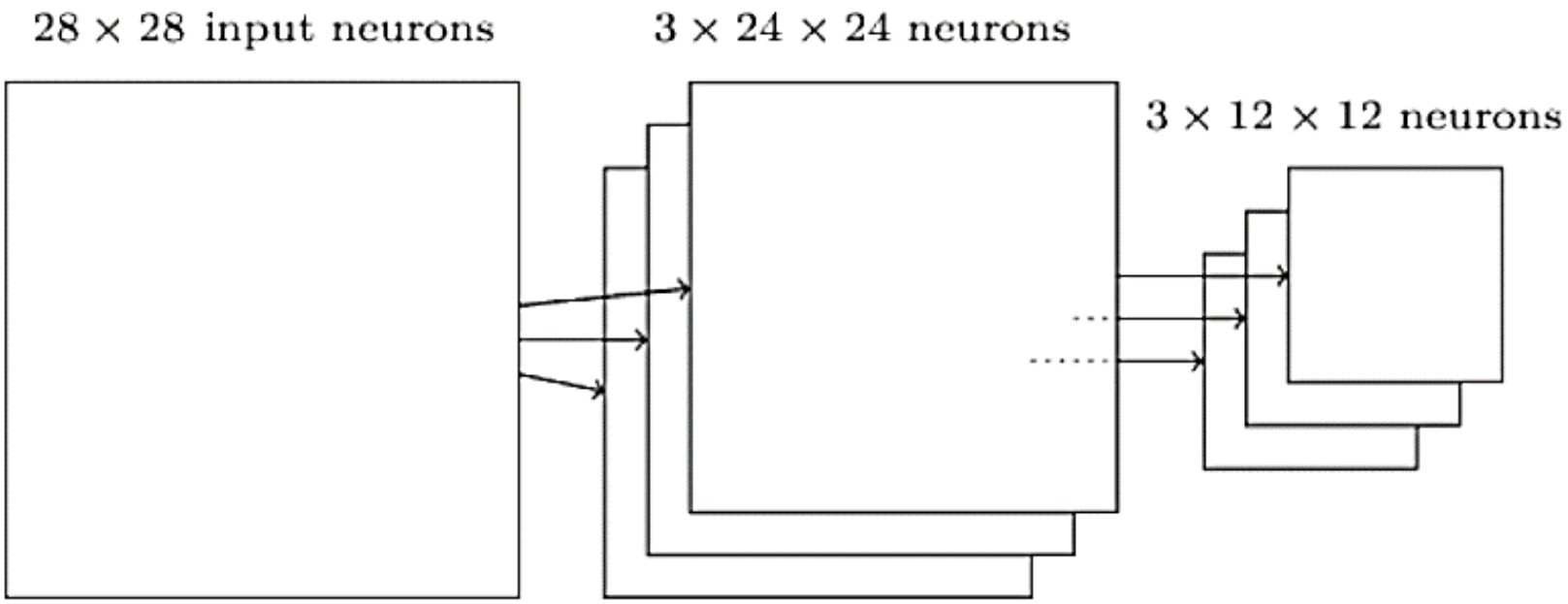
Pooling

- Tầng pooling nên để ngay sau tầng convolution
- Ánh xạ một vùng trên lớp trước thành 1 neural ở lớp sau bởi một hàm giản đơn nào đó (ví dụ: hàm max, hàm average,…)



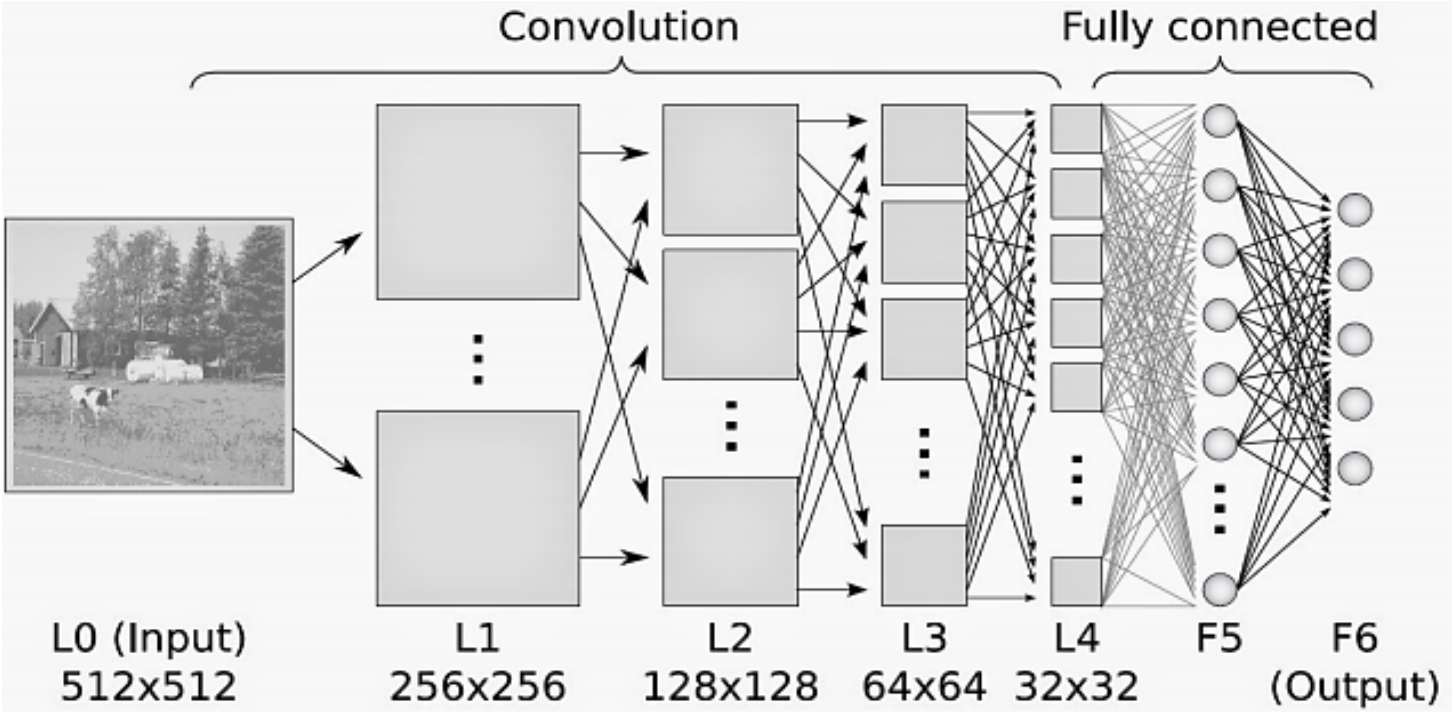
Pooling

- Lớp pooling cho phép giải quyết vấn đề sai lệch (nhỏ) của các feature
- Giảm kích cỡ của mạng
- Chú ý: max pooling luôn là tốt nhất



Các tầng khác

- Các lớp sau cùng của mạng thường là các lớp ANN thông thường, kết nối đầy đủ
- Riêng lớp cuối cùng có thể là softmax nếu bài toán là loại phân lớp xác suất



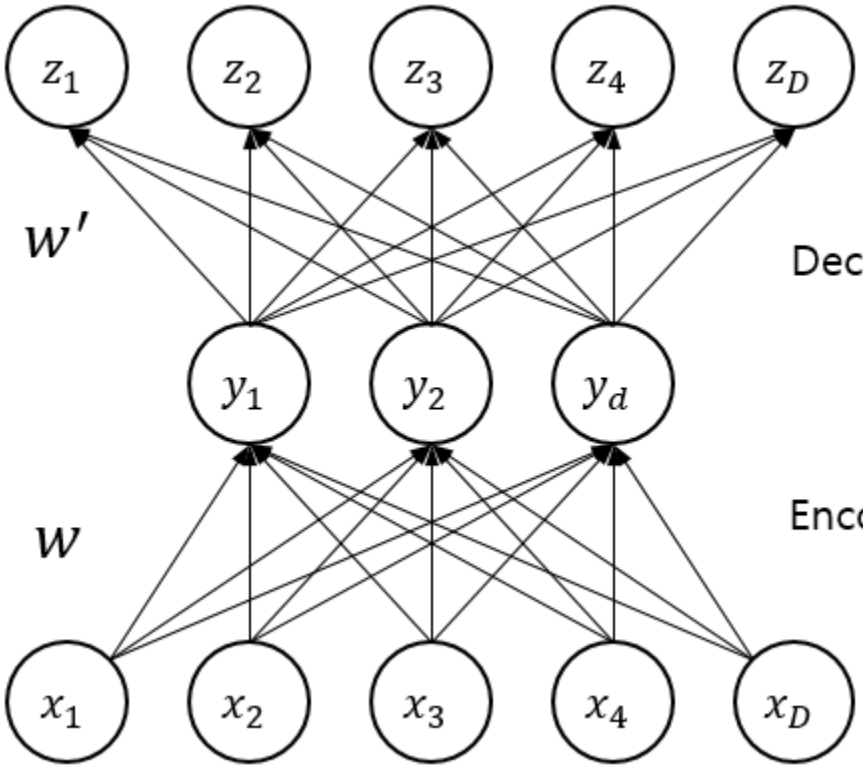
Alex-net

- Có model pre-trained sẵn, có thể tải về để sử dụng vào những mục đích khác

Phần 4

Bộ tự mã hóa (autoencoder)

Bộ tự mã hóa

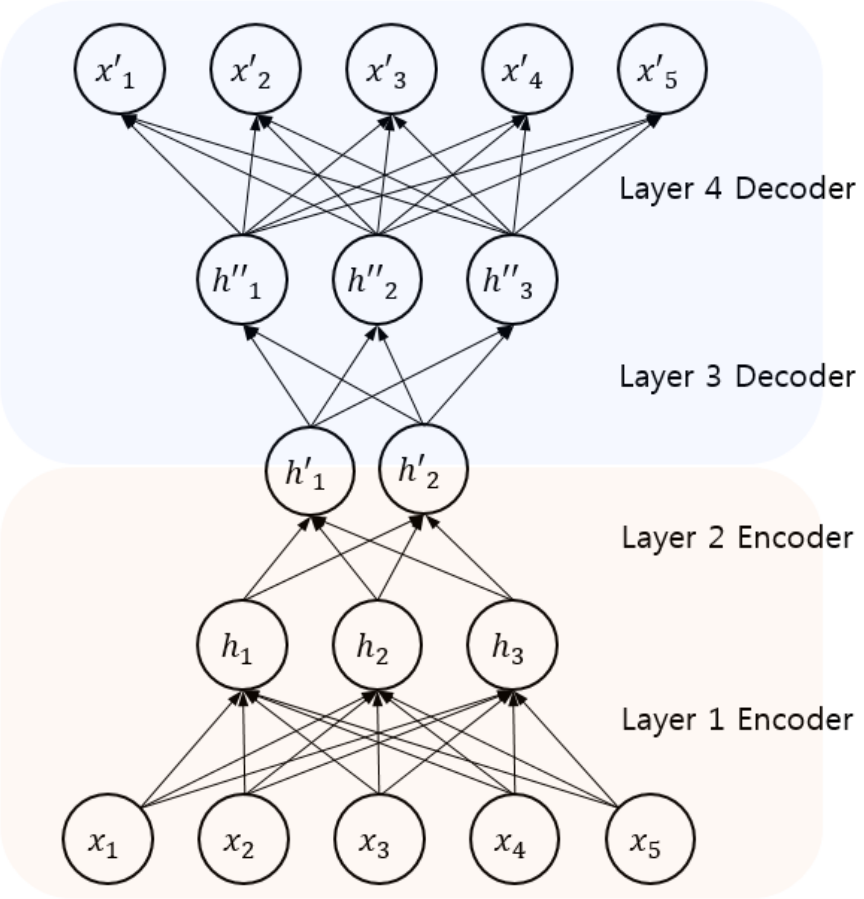


Decoder : $z = g_{\theta'}(y) = s(W'y + b')$

Encoder : $y = f_\theta(x) = s(Wx + b)$

Bộ tự mã hóa

- Các layer h, h’ và h’’ thực chất là sự mã hóa của dữ liệu đầu vào
- Nói cách khác, hệ thống tự ánh xạ các mẫu trong không gian n chiều đầu vào sang một không gian có số chiều tùy chọn
- Kĩ thuật này cho phép giảm số chiều một cách hiệu quả



Phần 5

Bắt đầu với học sâu như thế nào?

Tìm hiểu

- Học và hiểu mọi thứ trên giấy
  - Hoạt động của mạng: truyền thẳng (feed forward), tái phát (recurrent),…
  - Các thuật toán tập huấn: back-propagation, SGD,…
  - Ý nghĩa của các siêu tham số
  - Các kĩ thuật hiệu chỉnh mạng (khó)
- Đừng cố lập trình
  - Rất phức tạp
  - Rất dễ sai và khó phát hiện
  - Tốn nhiều thời gian
  - Không tái sử dụng được
- Trực quan: https://playground.tensorflow.org

Lập trình

- Sử dụng các framework có sẵn
- Mỗi framework có hay dở riêng
  - Chọn cái nào phù hợp với khả năng lập trình hiện tại
  - Không cầu toàn, nếu sau một thời gian thấy không hợp thì chuyển qua cái khác
    - Dữ liệu hầu như chia sẻ được giữa các framework
    - Chỉ phải lập trình lại
- Hầu hết sử dụng python, các tài liệu cũng vậy
- Đề xuất:
  - Theano (python)
  - TensorFlow (python)

Lập trình

- Nếu không dùng python có thể cân nhắc:
  - Caffe (C++)
  - CNTK (C#)
  - Torch (Lua)
  - Deeplearning4j (java)
- Chú ý: ngoại trừ Theano viết bằng python, các framework khác chủ yếu viết bằng C++
- Nên bắt đầu với một bài toán cụ thể và đủ nhỏ, chẳng hạn MNIST hoặc Iris
- Có GPU: tăng tốc độ huấn luyện gấp nhiều lần