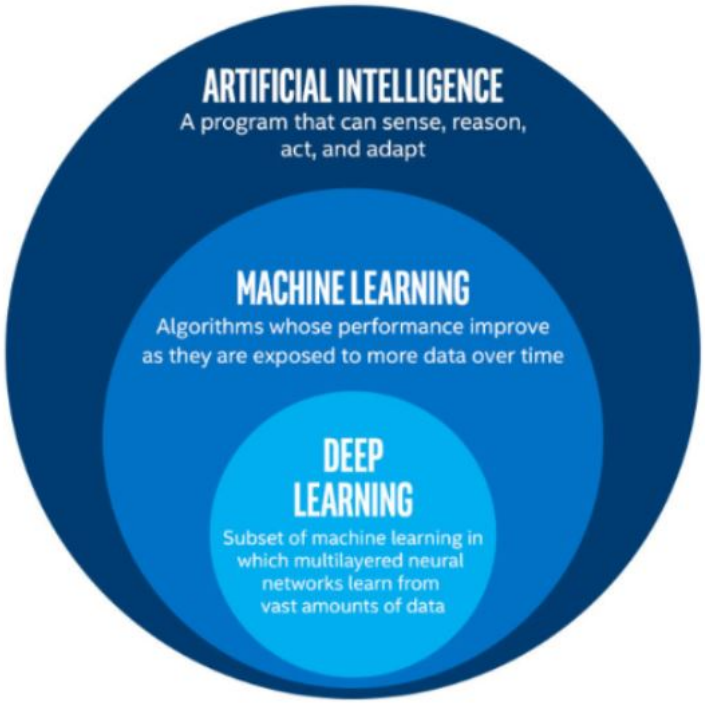
Deep learning và neural network

1. Giới thiệu chung về deep learning

Deep learning là một lĩnh vực con của Machine learning. Tuy nhiên, trong khoảng 10 năm trở lại đây, Deep learning đã có những bước chuyển mình mạnh mẽ để mang lại rất nhiều những ứng dụng thực tiễn dành cho người dùng.



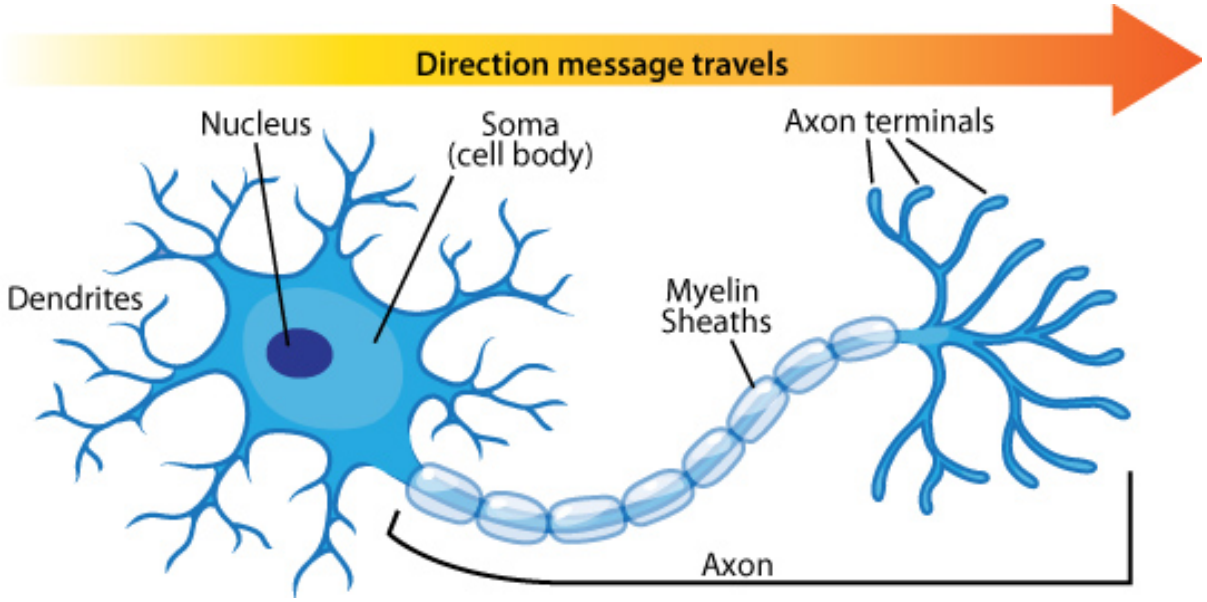
AI vs. ML. vs DL

Hai lý do chính dẫn đến sự phát triển vượt bậc của Deep learning trong những năm trở lại đây:

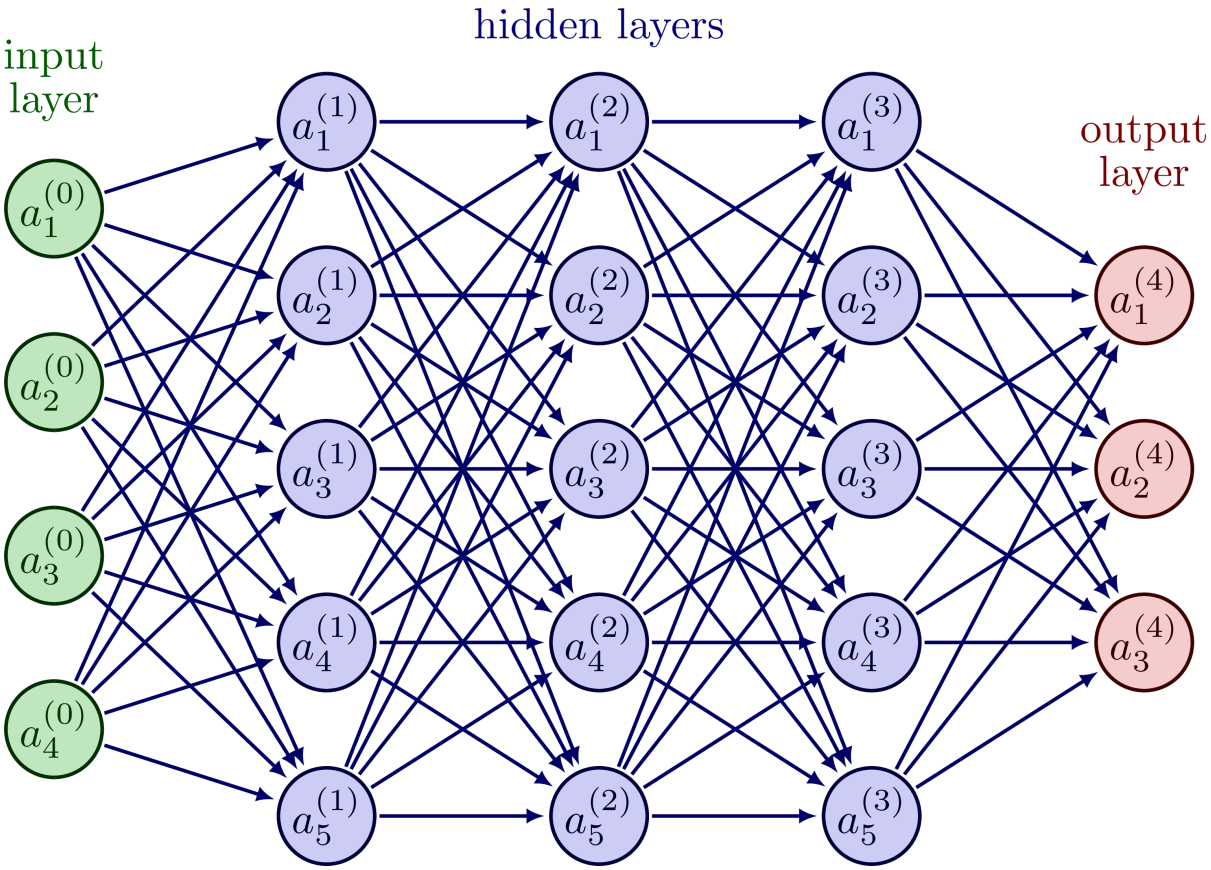
- Sự bùng nổ của dữ liệu: càng ngày chúng ta càng có nhiều dữ liệu được sinh ra với rất nhiều các thể loại khác nhau và các mô hình deep learning thì luôn khai thác rất tốt sự dồi dào của dữ liệu.
- Sự phát triển của phần cứng máy tính: song hành với lượng lớn dữ liệu, các mô hình deep learning cũng cần có hệ thống phần cứng mạnh mẽ để sẵn sàng huấn luyện hàng tỷ dữ liệu trong thời gian dài.

2. Kiến trúc mạng nơ ron - Neural network

Được lấy cảm hứng từ mạng nơ ron sinh học trong bộ não của con người, neural network là kiến trúc mạng gồm rất nhiều các nơ ron kết nối với nhau theo một trật tự nhất định.



Các nơ ron trong neural network được kết nối với nhau tạo thành các lớp (layer). Ta có lớp nhận đầu vào được gọi là input layer, các lớp tính toán ở giữa được gọi là hidden layer, lớp trả đầu ra được gọi là output layer.



Trong neural network có rất nhiều các loại layer khác nhau, có các chức năng khác nhau. Trong nội dung bài này, chúng ta sẽ nghiên cứu về một số các loại layer cơ bản nhất.

2.1. Linear layer

Linear layer là layer đơn giản nhất nhưng nền tảng để xây dựng neural network. Cụ thể, linear layer thực hiện phép biến đổi tuyến tính (linear transformation) thông qua phép toán nhân ma trận.

$$y = WX$$

trong đó:

- X là ma trận đầu vào của linear layer

- W là trọng số của neural network tại linear layer đó
- y là ma trận đầu ra của linear layer, kết quả của phép biến đổi tuyến tính

2.2. Activation layer

Chúng ta đã cùng nhau nghiên cứu về một số các logistic activation layer như Sigmoid, Softmax, Tanh. Bên cạnh các layer đó, còn rất nhiều các activation layer khác. Các activation layer được đặt xen kẽ giữa các linear layer với vai trò giúp cho các linear layer có nghĩa. Điều này đồng nghĩa với việc, nếu không có các activation layer đặt giữa các linear layer thì nhiều các linear layer đặt chồng lên nhau cũng không khác gì so với việc chỉ có một linear layer.

2.2.1. ReLU

Khác với Sigmoid hay Softmax thường dược đặt ở layer cuối cùng trong neural network, ReLU (Rectified-Linear Unit) là một activation layer thường được đặt giữa các linear layer nằm giữa của neural network.

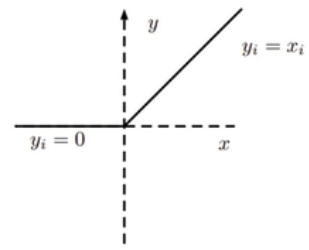
$$y = max(0, x)$$

trong đó:

- x là giá trị đầu vào của hàm ReLU
- y là giá trị đầu ra của hàm ReLU

Khi sử dụng hàm ReLU cho một vector hoặc ma trận, ta sử dụng hàm ReLU cho từng phần tử trên vector hay ma trận.

16	24	-13	7
17	-91	5	100
31	-5	43	-19
122	73	38	23



ReLU Layer

16	24	0	7
17	0	5	100
31	0	43	0
122	73	38	23

2.2.2. Leaky ReLU

Một vấn đề khi sử dụng hàm ReLU là vanishing gradient tại những vị trí có giá trị nhỏ hơn hoặc bằng 0. Do đó, Leaky ReLU giúp cải thiện vấn đề này.

$$y = max(\gamma x, x)$$

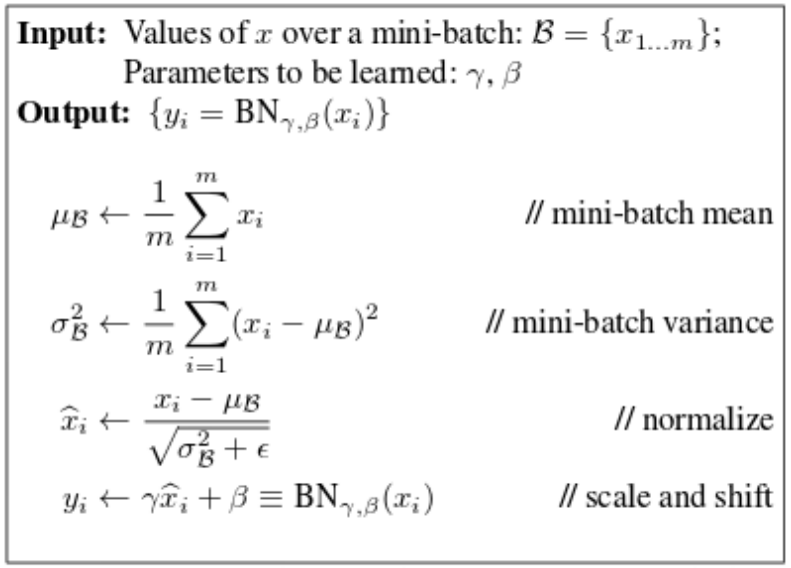
trong đó:

- x là giá trị đầu vào của hàm ReLU
- γ là giá trị rất nhỏ, thường được lựa chọn là 0.1
- y là giá trị đầu ra của hàm ReLU

2.3. Normalization layer

Đối với các neural network các phức tạp và có kích thước mô hình lớn, quá trình huấn luyện càng khó khăn và bất ổn định. Sự bất ổn định có thể dẫn đến việc neural network có kết quả huấn luyện rất kém.

Sự ra đời của các Normalization layer, và cụ thể là Batch normalization layer đã giúp cải thiện đáng kể tình trạng này. Các normalization layer nói chung giúp chuẩn hoá đầu ra của mỗi layer trong neural network từ đó giúp ổn định hoá quá trình huấn luyện.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

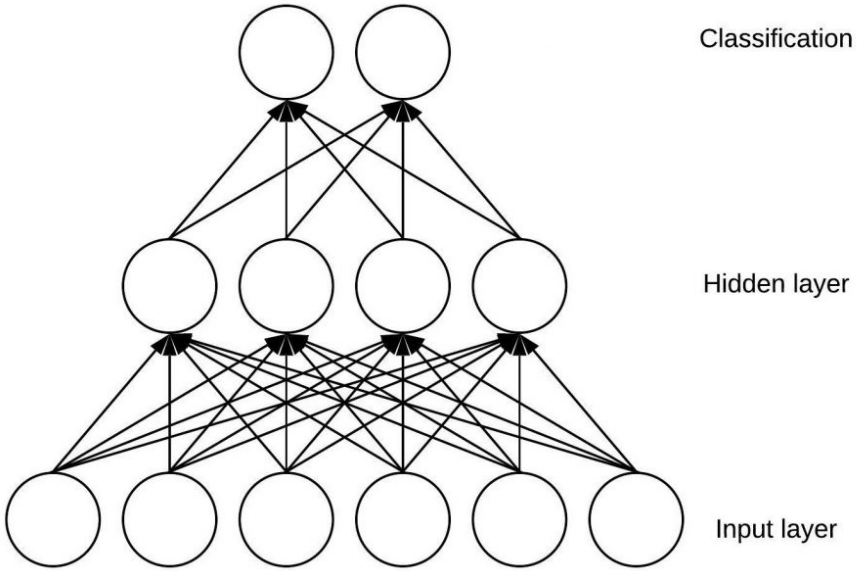
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Bên cạnh Batch normalization layer, ta còn một số các normalization layer khác như Layer normalization layer, Instance normalization layer, Group normalization layer ...

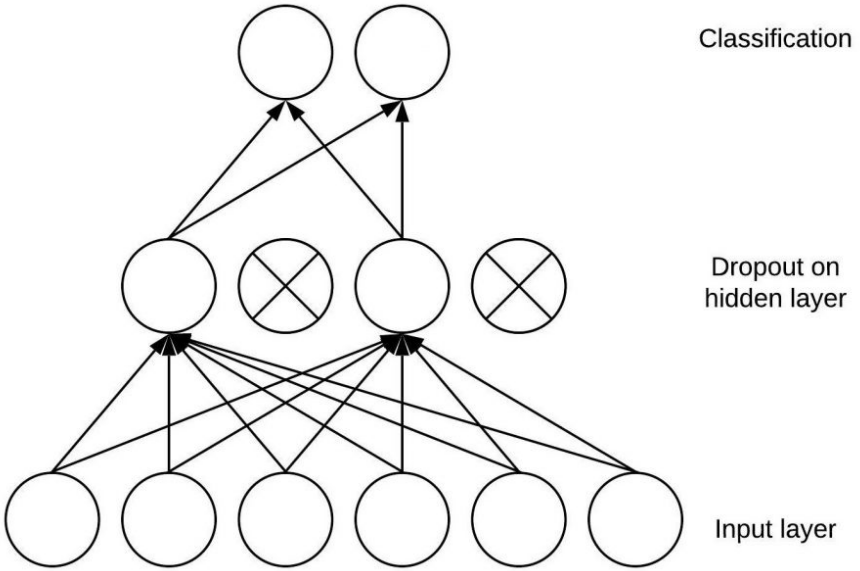
2.4. Dropout layer

Chúng ta đã bàn luận về hiện tượng overfit trong các mô hình machine learning, trong neural network, hiện tượng overfit cũng là một vấn đề nan giải. Ta có Dropout layer là một lớp giúp phần nào đó giảm bớt vấn đề overfit.

Cụ thể, dropout layer sẽ ngẫu nhiên lựa chọn một số các nơ ron trong neural network, chính xác hơn là lựa chọn ngẫu nhiên một số các trọng số của mô hình và gán giá trị bằng 0. Từ đó, ta sẽ giảm được độ phức tạp của neural network, từ đó giảm được hiện tượng overfit.



Without Dropout



With Dropout

3. Loss function và Optimization

3.1. Loss function

Chúng ta đã cùng nhau nghiên cứu về khá nhiều các loại hàm loss khác nhau và cụ thể cho từng bài toán khác nhau.

Đối với bài toán Regression ta có một số hàm loss như

- Mean absolute error

$$MAE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$$

- Mean squared error

$$MSE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Đối với bài toán Classification ta có một số hàm loss như

- Binary cross entropy

$$BCE(\hat{y}, y) = -\sum_{i=1}^{N}(y^i \log \hat{y}^i + (1 - y^i) \log(1 - \hat{y}^i))$$

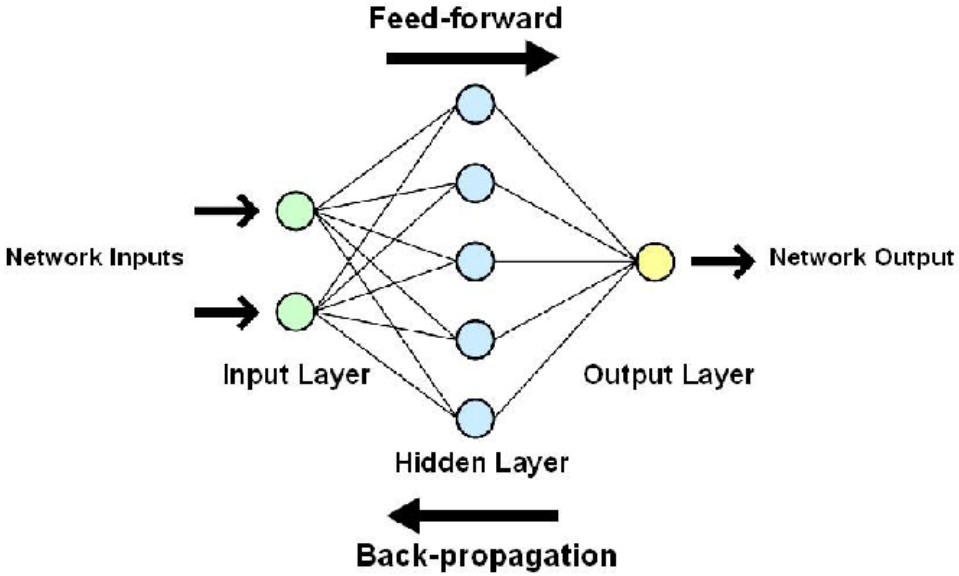
- Categorical cross entropy hay cross entropy

$$CE(\hat{y}, y) = -\sum_{i=1}^{N}\sum_{j=1}^{K}(y^{ij} \log \hat{y}^{ij})$$

3.2. Optimization

Chúng ta đã cùng nhau nghiên cứu về khá nhiều các thuật toán tối ưu khác nhau biến thể từ Gradient descent như: Stochastic gradient descent (SGD), Momentum, Nesterov accelerated gradient (NAG) ...

4. Model feedforward và backpropagation



Feedforward là quá trình neural network nhận đầu vào input layer, thực hiện các phép tính toán qua các linear layer, activation layer, normalization layer ... để trả đầu ra output layer.

Ngược lại với feedforward, sau khi đưa kết quả dự đoán của neural network vào hàm loss và tính toán giá trị loss, Backpropagation là quá trình tính toán giá trị đạo hàm của hàm loss theo từng trọng số của neural network. Sau khi tính được giá trị gradient tương ứng với mỗi trọng số, neural network thực hiện cập nhật lại các trọng số này theo thuật toán tối ưu dựa trên gradient descent.