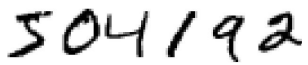
Chương 7
Multi-layer Perceptron

TS. Phạm Tuấn

ptuan@ute.udn.vn

Giới thiệu

Hệ thống thị giác của con người là một trong những kỳ quan của thế giới. Hãy xem xét dãy chữ số viết tay sau:



Hầu hết mọi người dễ dàng nhận ra những chữ số đó là 504192. Sự dễ dàng ở đây là cả quá trình xử lý phức tạp ở não con người. Trong mỗi bán cầu não của chúng ta, nhiều hơn 140 triệu tế bào thần kinh, với hàng chục tỷ kết nối giữa chúng để thực hiện các xử lý phức tạp về hình ảnh.

Giới thiệu

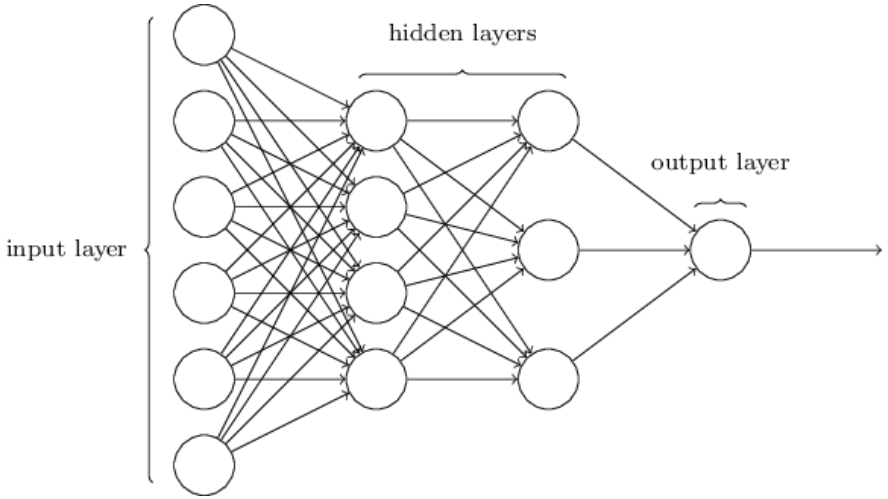
Khó khăn trong việc nhận dạng mẫu trực quan trở nên rõ ràng nếu bạn cố gắng viết một chương trình máy tính để nhận dạng các chữ số như những chữ số ở trên. Điều tưởng chừng dễ dàng khi chúng ta tự mình làm lại bỗng trở nên vô cùng khó khăn.

Những trực giác đơn giản về cách chúng ta nhận ra hình dạng - "số 9 có một đường tròn ở trên cùng và một nét dọc ở dưới cùng bên phải" - hóa ra không đơn giản để diễn đạt theo thuật toán.

Khi bạn cố gắng thực hiện chính xác các quy tắc như vậy, bạn sẽ nhanh chóng bị lạc vào một đống các ngoại lệ, cảnh báo và các trường hợp đặc biệt. Nó dường như vô vọng.

Perceptron (Artificial Neuron)

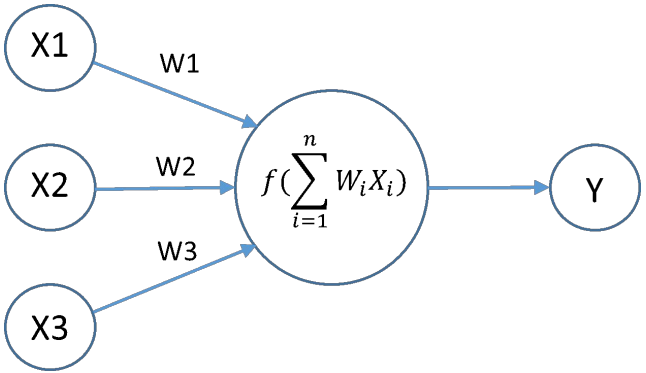
Thay vì chúng ta tạo ra một bộ quy tắc để nhận diện chữ số, chúng ta sẽ tạo ra một mô hình mô phỏng hoạt động của não người. Để làm được điều đó, một neuron nhân tạo (Perceptron) thay cho tế bào thần kinh của con người, và giữa các neuron nhân tạo được liên kết với nhau.



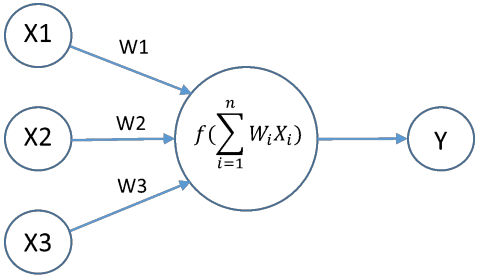
Perceptron (Artificial Neuron)

Bây giờ chúng ta đến với một ví dụ đơn giản của Perceptron. Một perceptron nhận một số đầu vào nhị phân, x1, x2, x3 và tạo ra một đầu ra nhị phân duy nhất. Và hàm *f* ở đây được gọi là **activation function**.

Chúng ta đưa ra các trọng số, w1, w2,..., các số thực để thể hiện tầm quan trọng của các đầu vào tương ứng đối với đầu ra



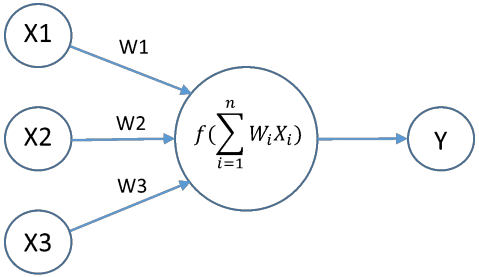
Perceptron (Artificial Neuron)



Đầu ra của Neuron là, 0 hoặc 1, được xác định bằng cách xem giá trị của **hàm activation** nhỏ hơn hoặc lớn hơn một **giá trị ngưỡng nào đó**. Cũng giống như trọng số, ngưỡng là một số thực, là một tham số của Neuron.

$$output = \begin{cases} 1 & if \ f(wx) > threshold \\ 0 & if \ f(wx) \leq threshold \end{cases}$$

Perceptron (Artificial Neuron)

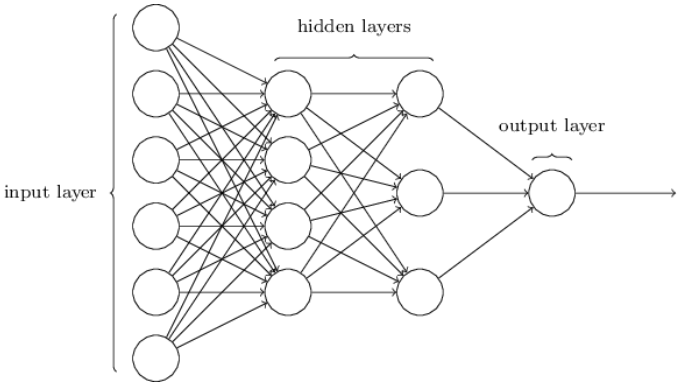


Giả sử sắp đến cuối tuần và bạn nghe nói rằng sắp có buổi ca nhạc. Bạn đang cố quyết định xem có nên tham gia lễ hội hay không. Bạn có thể đưa ra quyết định của mình bằng cách cân nhắc ba yếu tố:

1. Trời có mưa hay không (x1 = 1, w1 = 0.2)
2. Có người đi cùng với bạn hay không (x2 = 0, w2 = 0.4)
3. Giá vé (x3 = 1, w3 = 0.4)

Multi-layer Perceptron (MLP)

Chúng ta quay trở lại với tiêu đề của bài học là Multi-layer Perceptron, tức là một mạng lưới gồm nhiều lớp (**Layer**) và trong mỗi layer lại có nhiều **unit** (Perceptron). Nhưng ví dụ vừa rồi, chúng ta có thể thấy sự tương tự giữa Perceptron và Logistic regression, tuy nhiên MLP mạnh mẽ hơn Logistic regression rất nhiều. **Đặc biệt MLP có thể học theo bất cứ hàm nào**. Và Multi-layer perceptron hay còn được gọi là Deep neural networks.



MLP vs Logistic regression

Logistic regression	MLP
Là một classifier	Là một classifier và predictor
Đầu vào là những features	Đầu vào là dữ liệu nguyên bản hoặc raw features
Mô hình đơn giản	Mô hình phức tạp với nhiều layers và units
Dữ liệu đơn giản	Dữ liệu nhiều và phức tạp

Units

Cho một tập hợp các điểm đầu vào $x_1 \ldots x_n$, một unit có các trọng số tương ứng $w_1 \ldots w_n$ and một bias b. Và tổng trọng số của z có thể biểu diễn như sau:

$$z = b + \sum_i w_i x_i$$

Để thuận tiện hơn khi thể hiện tổng trọng số này bằng cách sử dụng ký hiệu vectơ, chúng ta thay thế ∑ bằng dot product:

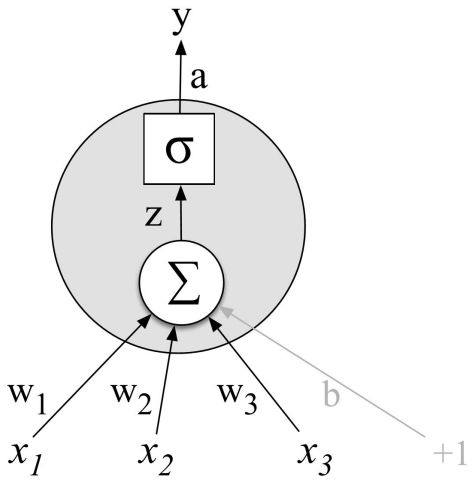
$$z = w \cdot x + b$$

Và một hàm số phi tuyến tính f được áp vào z. Hàm phi tuyến ở đây là hàm **activation**, và sigmoid là một trong số đó:

$$y = \sigma(z) = \frac{1}{1 + e^{(-z)}} \quad \Rightarrow \quad y = \sigma(w \cdot x + b) = \frac{1}{1 + exp(-(w \cdot x + b))}$$

Unit

Hình vẽ bên dưới hiển thị một giản đồ cuối cùng của Unit cơ bản. Trong ví dụ này, Unit nhận 3 giá trị đầu vào x_1, x_2 và x_3 và tính tổng có trọng số, nhân từng giá trị giá trị theo trọng số (tương ứng w_1, w_2 và w_3), thêm chúng vào một số bias b, và sau đó chuyển tổng kết quả qua một hàm sigmoid để được một số giữa 0 và 1.



Activation Function

Có 4 hàm activation chính sau:

1. sigmoid
2. tanh
3. ReLU
4. softmax

[0, 1]

$$sigmoid(z) = \frac{1}{1+e^{(-z)}} \qquad \frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

[0, 1]

$$softmax(z_i) = \frac{e^{z_i}}{\sum_j^d (e^{z_j})}$$

[-1, 1]

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \qquad \frac{d\tanh(z)}{dz} = 1-\tanh^2(z)$$

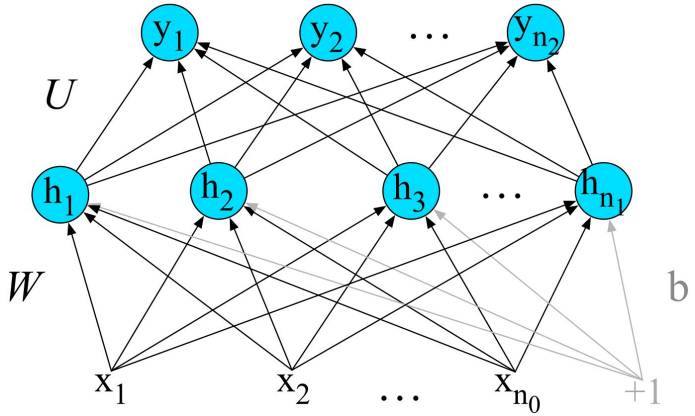
[0, +∞]

$$ReLU(z) = max(z, 0) \qquad \frac{d\,\mathrm{ReLU}(z)}{dz} = \begin{cases} 0 & for \ x < 0 \\ 1 & for \ x \geq 0 \end{cases}$$

Multi-layer Perceptron

Một mạng MLP sẽ có 3 loại nodes sau: input units, hidden units, and output units.

Cốt lõi của MLP là lớp ẩn được tạo thành từ các hidden units, mỗi Unit tương ứng với một neuron, ta lấy tổng trọng số các đầu vào của nó và sau đó áp dụng các hàm tính phi tuyến tính để được đầu ra. Trong kiến trúc tiêu chuẩn, mỗi lớp là **fully-connected** với lớp trước và sau nó.



Hidden layer

Đầu ra của các hidden layer là vector h, vector h được tính như sau:

$$h = \sigma(Wx + b)$$

Từ phương trình trên chúng ta có những chú ý như sau: hàm activation áp dụng trên từng phần tử.

Nếu chúng ta coi w_0 là b và chúng ta thêm vào 1 dummy unit với giá trị bằng 1 thì công thức trên được viết lại như sau:

$$h = \sigma(Wx)$$

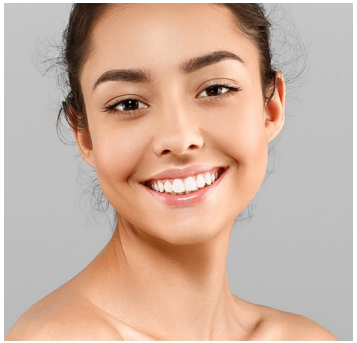
Chúng ta diễn giải công thức trên như sau:

$$h_j = \sigma\left(\sum_{i=0}^{n} W_{ji} x_i\right)$$

Hidden layer

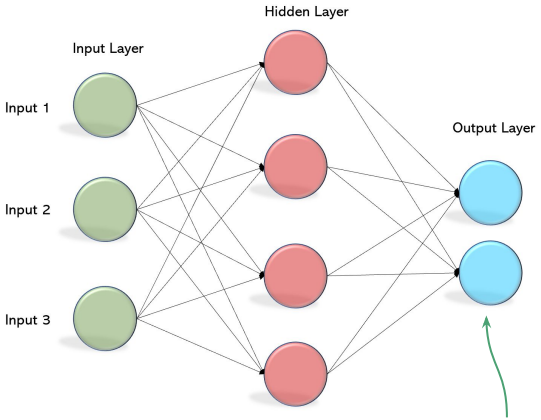
Hidden layer được coi là những lớp ẩn, và nó chính là đại diện hoặc đặc tính của dữ liệu đầu vào. Vai trò của lớp đầu ra là nhận dữ liệu từ các lớp hidden layer và tính toán kết quả cuối cùng. Đầu ra này có thể là một số có giá trị thực.

Chúng ta lấy một ví dụ như sau về. Nếu coi bức ảnh khuôn mặt là dữ liệu đầu vào, thì hidden layer sẽ chứa những đặc tính của khuôn mặt.

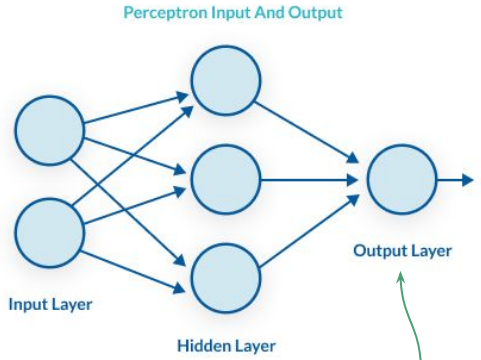


Output layer

Output layer là layer cuối cùng của mạng MLP. Tùy thuộc bài toán phân loại hoặc hồi quy, thì số lượng unit trong output layer là khác nhau.



Phân loại softmax

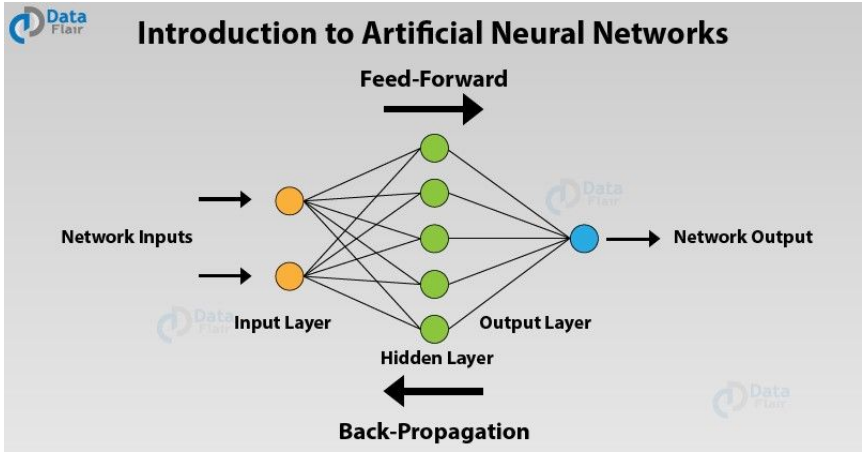


Hồi quy sigmoid

Data flow

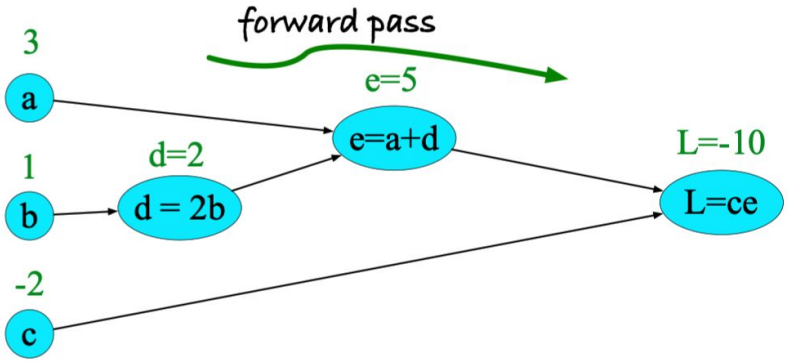
Dòng dữ liệu của MLP sẽ gồm 2 quá trình: feed forward và backward propagation. trong đó:

1. Feed forward sẽ đưa ra giá trị đầu ra của mô hình mạng MLP.
2. Backward propagation sẽ cập nhật các thông số(?) của mô hình mạng MLP bằng Gradient descent.



Forward pass

Để hiểu forward pass, chúng ta làm ví dụ như sau với hàm số $L(a,b,c) = c(a+2b)$. với giá trị đầu vào vào là a = 3, b = 1, c = -2

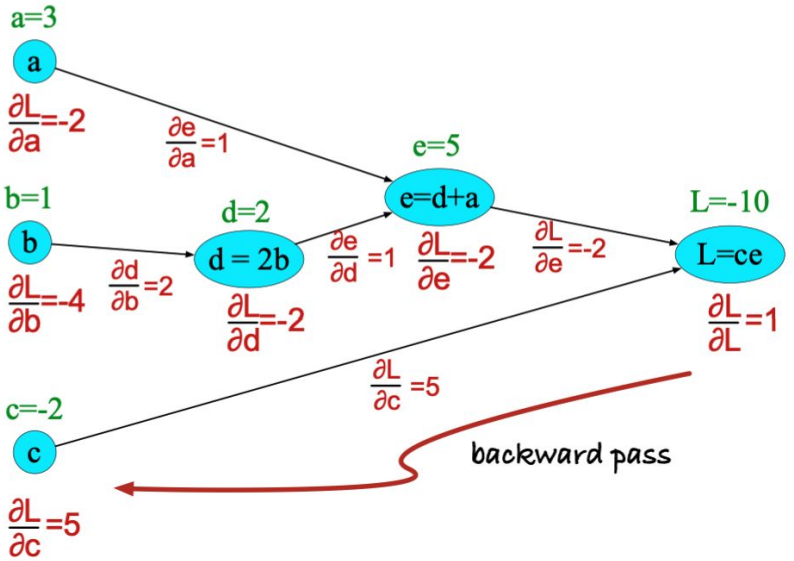


Backward propagation

Quá trình Backward bao gồm đến **tính đạo hàm của hàm loss tại unit** của mô hình MLP. Chúng ta dùng Chain rule để tính đạo hàm tại các node.

Ta lấy ví dụ tính đạo hàm của hàm tạo node **b**. Hàm **L** muốn đến được **b** phải thông qua **e, d** nên chúng ta tính như sau:

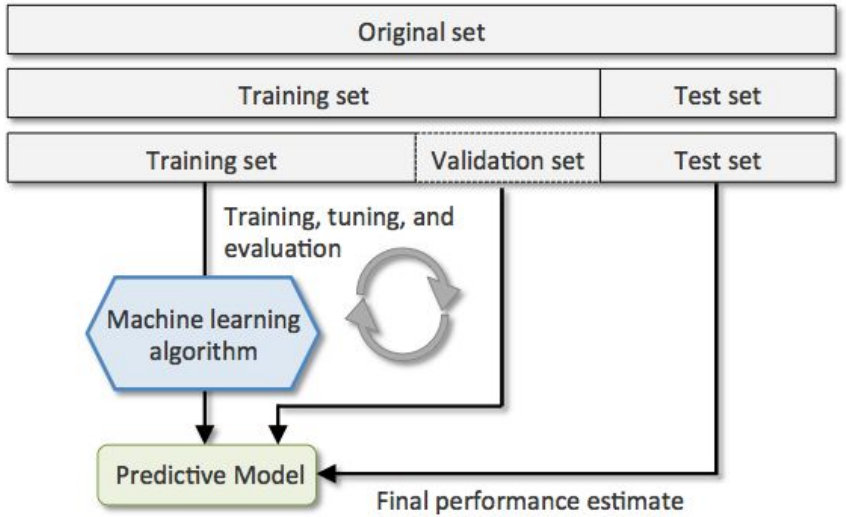
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial b} = c \cdot 1 \cdot 2 = -4$$



Thiết lập quá trình huấn luyện

Thiết lập quá trình huấn luyện sẽ có những phần sau:

- Tạo bộ dataset gồm 3 phần (Training set, Validation set và Test set).
- Xây dựng mô hình.
- Bắt đầu quá trình training.



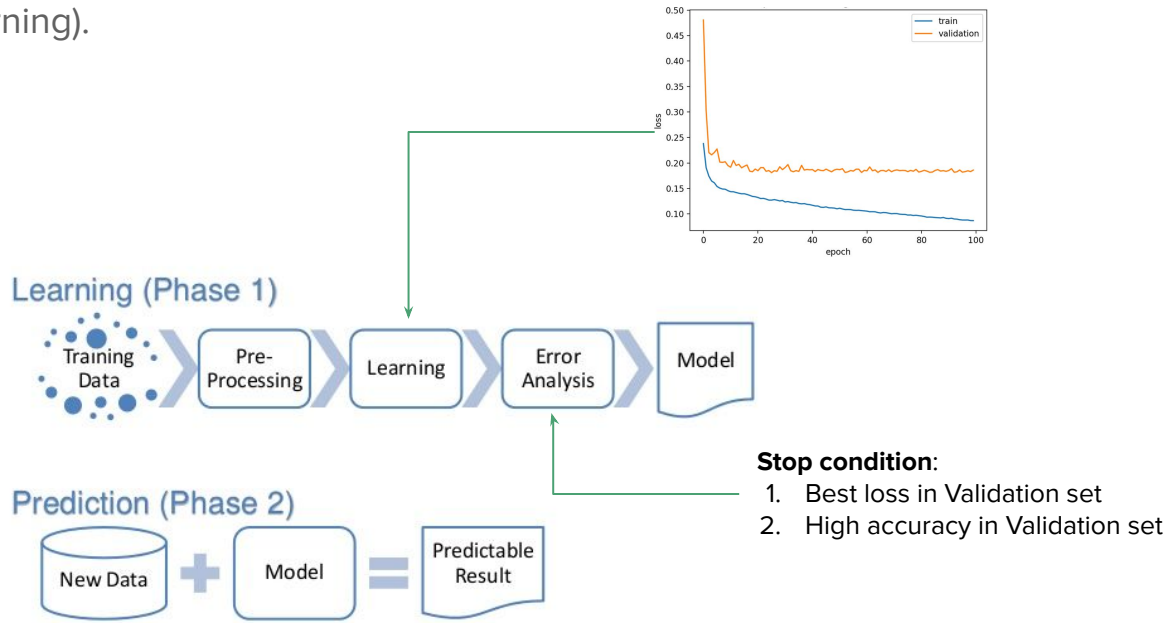
Xây dựng mô hình - Hyperparameter (tham số)

Điều chỉnh hyperparameters cũng rất quan trọng. Các tham số của mạng MLP là trọng số W và bias b; chúng được học bằng cách dùng Gradient descent. **Còn các hyperparameter là những thứ được chọn bởi người phát triển mô hình**. Các giá trị tối ưu được điều chỉnh trên một tập dữ liệu gọi là **validation set** chứ không phải bằng cách dùng Gradient descent trên training set. **Hyperparameter** bao gồm t**ốc độ học α**, kích thước **batch size**, **kiến trúc mô hình** (số của các layer, số lượng các hidden unit trên mỗi lớp, sự lựa chọn các **activation function**)...

Quá trình tạo ra mô hình MLP

Quá trình tạo ra một mô hình Deep learning nói chung và MLP nói riêng sẽ thường có 2 bước:

1. Training (learning).
2. Prediction.



Stop condition:

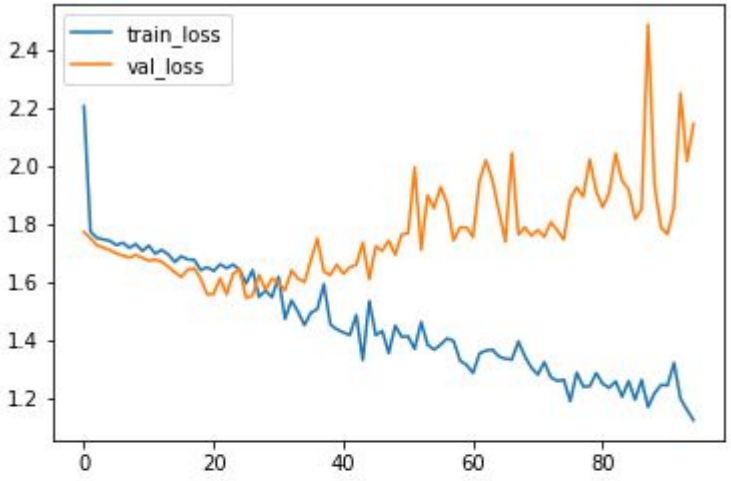
1. Best loss in Validation set
2. High accuracy in Validation set

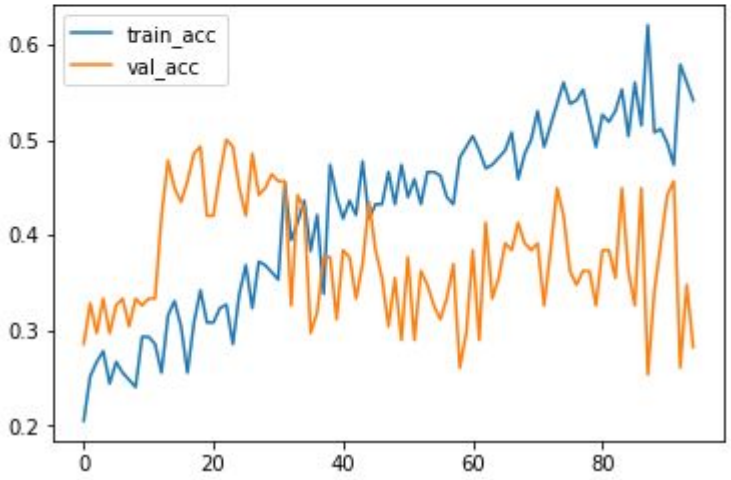
Overfit

Overfit là hiện tượng học quá khớp, có nghĩa là mô hình MLP đang quá khớp với training data, và sẽ thể hiện hiệu suất thấp ở tập validation set cũng như testing set.

Overfit báo rằng mô hình MLP chúng ta đang xây dựng là quá lớn, hoặc quá trình train đã xảy ra quá lâu. (Chú ý rằng mô hình càng lớn thì quá trình xảy ra hiện tượng overfit càng nhanh, do mô hình lớn thì khả năng học từ dữ liệu nhanh hơn so với mô hình nhỏ)

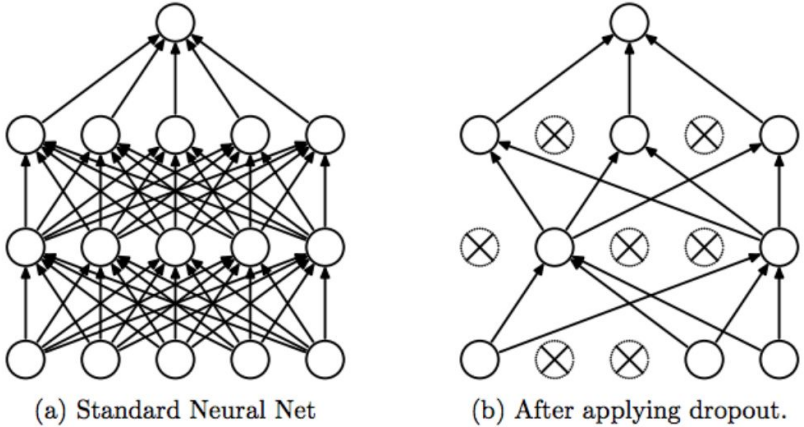
Để xây dựng một mô hình MLP tốt thì chúng tăng từ số hidden layer cũng như hidden unit. Tới khi nào *validation error* có chiều hướng tăng lên thì chúng ta bắt đầu đã đạt tới ngưỡng giới hạn





Dropout

Khi mô hình MLP càng lớn so với data thì hiện tượng overfit càng dễ xảy, tuy nhiên mô hình MLP càng lớn tức là càng phức tạp thì hiệu suất dự đoán cũng cao hơn. Khi chúng ta có một vấn đề như sau: **Chúng ta muốn xài một mô hình MLP lớn để đạt hiệu suất cao nhưng phải tránh được overfit**. Dropout chính là một technique làm giảm hiện tượng overfit.



(a) Standard Neural Net

(b) After applying dropout.

Any questions ?