Logistic regression

1. Ý tưởng chung của logistic regresion

Mô hình Linear regression sử dụng phép biến đổi tuyến tính, trả đầu ra nằm trong khoảng $[-\infty, \infty]$ phù hợp cho việc giải quyết bài toán regression.

$$\hat{y} = WX$$

Kế thừa điều này, mô hình Logistic regression biến đổi đầu ra của phép biến đổi tuyến tính sao cho có thể sử dụng được nó để giải quyết bài toán classification. Cụ thể, Logistic regression áp dụng một hàm số f lên đầu ra của phép biến đổi tuyến tính để thu được đầu ra như mong muốn.

$$\hat{y} = f(WX)$$

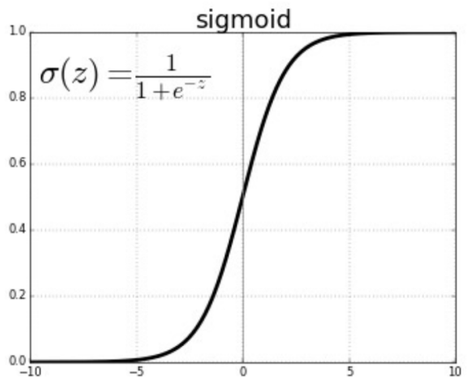
2. Các hàm logistic activation (logistic activation function)

Các hàm số f như trên được gọi là các hàm logistic activation.

Có một số các hàm logistic activation phổ biến là Sigmoid, Tanh và Softmax.

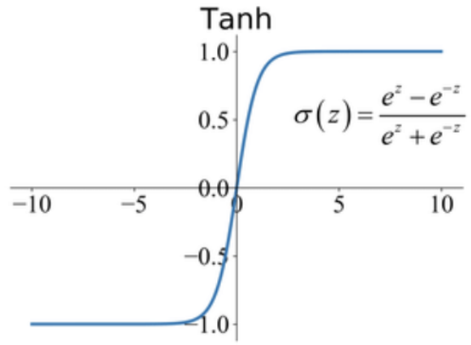
2.1. Sigmoid

Hàm sigmoid nhận đầu vào là bất kỳ giá trị nào trong khoảng $[-\infty, \infty]$ và trả đầu ra nằm trong khoảng trong khoảng $[0, 1]$ và giá trị đầu ra này có thể được hiểu như giá trị xác suất.



2.2. Tanh

Hàm tanh nhận đầu vào là bất kỳ giá trị nào trong khoảng $[-\infty, \infty]$ nhưng khác với sigmoid, tanh trả đầu ra nằm trong khoảng trong khoảng $[-1, 1]$.

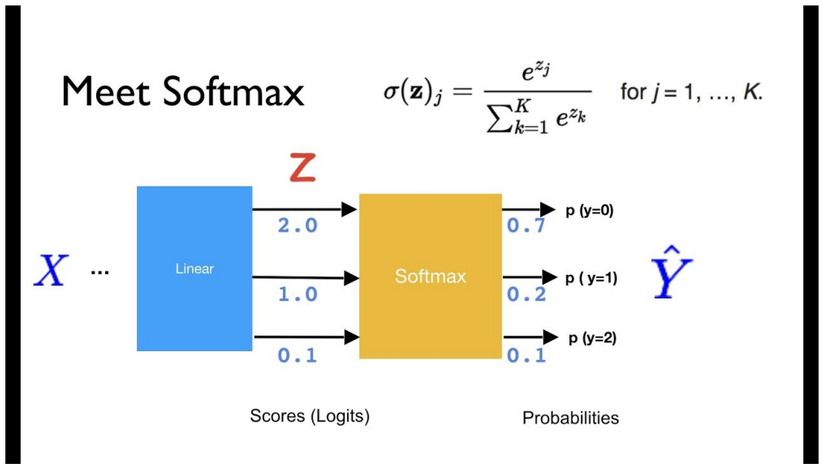


Ta có thể dễ dàng biến đổi khoảng giá trị đầu ra của hàm tanh từ $[-1, 1]$ về giống như sigmoid $[0, 1]$ thông qua công thức

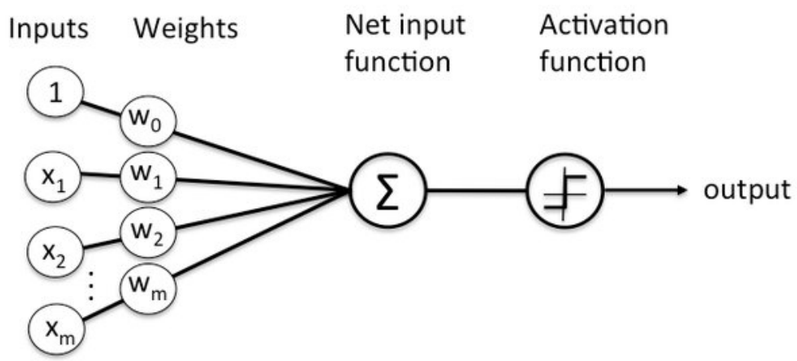
$$\tanh(z) = 2\sigma(2z) - 1$$

2.3. Softmax

Giống với hàm sigmoid, hàm softmax cũng trả đầu ra nằm trong khoảng $[0, 1]$, tuy nhiên thay vì nhận đầu vào chỉ một giá trị như sigmoid, softmax nhận đầu vào là một vector hôm nhiều giá trị.



3. Tối ưu mô hình logistic regression cho bài toán phân lớp nhị phân (binary classification)



Đối với mô hình logistic regression đơn giản, ta sử dụng hàm sigmoid làm activation function. Lúc này, bài toán phân lớn mà chúng ta giải quyết là bài toán phân lớp nhị phân (binary classification), cụ thể, mô hình sẽ được huấn luyện để phân biệt giữa hai lớp dữ liệu, lớp positive (được đại diện bởi số 1) và lớp negative (được đại diện bởi số 0).

Cụ thể hơn, với hàm sigmoid, mô hình sẽ tính toán và trả đầu ra là giá trị xác suất mà điểm dữ liệu thuộc lớp positive lớp số 1.

Giả sử, ta sử dụng bộ dữ liệu $X = [x^1, x^2, \dots, x^i]$. Xét điểm dữ liệu x^i, ta có $\sigma(Wx^i)$ là xác suất mà điểm dữ liệu x^i thuộc lớp số 1 và $1 - \sigma(Wx^i)$ là xác suất mà điểm dữ liệu x^i thuộc lớp số 0.

$$P(y^i = 1|x^i, W) = \sigma(Wx^i) = \hat{y}^i$$

$$P(y^i = 0|x^i, W) = 1 - \sigma(Wx^i) = 1 - \hat{y}^i$$

Kết hợp hai công thức trên, ta thu được

$$P(y^i|x^i, W) = (\hat{y}^i)^{y^i}(1 - \hat{y}^i)^{(1-y^i)}$$

Tính toán trên toàn bộ bộ dữ liệu

$$P(y|X, W) = \prod_{i=1}^{m} P(y^i|x^i, W) = \prod_{i=1}^{m} (\hat{y}^i)^{y^i}(1 - \hat{y}^i)^{(1-y^i)}$$

$$P(y|X, W) = (\hat{y})^{y}(1 - \hat{y})^{(1-y)}$$

Đến đây, mô hình cần học ra được giá trị W sao cho giá trị xác suất $P(y|X, W)$ là lớn nhất và tương đương, giá trị xác suất $-P(y|X, W)$ là nhỏ nhất.

$$W^* = \arg\max_W P(y|X, W) = \arg\min_W -P(y|X, W)$$

Với các giá trị xác suất đều nhỏ hơn 1, nên việc sử dụng phép nhân rất nhiều các giá trị nhỏ sẽ gây ra hiện tượng sai số trong máy tính. Do đó, ta sử dụng hàm logarit để biến đổi từ phép nhân thành phép cộng. Việc cực tiểu hoá giá trị $-P(y|X, W)$ tương đương với việc cực tiểu hoá $-\log P(y|X, W)$, từ đó, ta có hàm loss Binary Cross Entropy.

$$\mathcal{L}(W) = -\log P(y|X, W) = -\log\left(\hat{y}^y(1 - \hat{y})^{(1-y)}\right) = -(\log \hat{y}^y + \log(1 - \hat{y})^{(1-y)})$$

Biến đổi một chút,

$$\mathcal{L}(W) = -(\log \hat{y}^y + \log(1 - \hat{y})^{(1-y)}) = -(y \log \hat{y} + (1 - y)\log(1 - \hat{y})) = -\sum_{i}^{m}(y^i \log \hat{y}^i + (1 - y^i)\log(1 - \hat{y}^i))$$

Đến đây, để cực tiểu hoá giá trị hàm loss, ta sử dụng thuật toán gradient descent. Do đó, trước tiên, ta cần tính đạo hàm của hàm loss

$$\mathcal{L}(\hat{y}) = \mathcal{L}(\sigma(z)) = \mathcal{L}(\sigma(WX))$$

với

$$\hat{y} = \sigma(WX)$$

Ta có, theo chain rule,

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial W}$$

Đạo hàm của hàm loss binary cross entropy được tính như sau

$$\frac{\partial \mathcal{L}}{\partial \sigma} = \frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})}$$

Đạo hàm của hàm sigmoid được tính như sau

$$\frac{\partial \sigma}{\partial z} = \hat{y}(1 - \hat{y})$$

Đạo hàm của hàm biến đổi tuyến tính được tính như sau

$$\frac{\partial z}{\partial W} = X$$

Từ đó ta có

$$\frac{\partial \mathcal{L}}{\partial W} = (\hat{y} - y)X$$

Sau khi tính được đạo hàm của hàm loss theo W, ta áp dụng thuật toán gradient descent

$$W = W - \eta(\hat{y} - y)X$$

Ta cũng có thể sử dụng biến thể SGD của thuật toán gradient descent bằng cách cập nhật trọng số của mô hình với từng điểm dữ liệu

$$W = W - \eta(\hat{y}^i - y^i)x^i$$

4. Logistic regression giải quyết bài toán multi-label classification

Nếu như mô hình Logistic regression nói trên giúp giải quyết bài toán phân lớp nhị phân. Hay trong một ví dụ cụ thể về bài toán phân lớp ảnh chó hay mèo, mô hình logistic regression nói trên giúp ta trả lời câu hỏi "Trong ảnh có hình ảnh của chó hay không có hình ảnh của chó (nghĩa là có hình ảnh của mèo)???".

Tuy nhiên, trong một số trường hợp, với dữ liệu đầu vào có nhiều thông tin hơn, hay nói cách khác, ***dữ liệu đầu vào có thể có nhiều label***, ta phải giải quyết bài toán phân lớp nhiều label (multi-label classification). Ví dụ cụ thể, lúc này mô hình logistic regression cần trả lời giúp ta câu hỏi "Trong ảnh có hình ảnh của chó hay không có hình ảnh của chó?, có hình ảnh của mèo hay không có hình ảnh của mèo? có hình ảnh của gà hay không có hình ảnh của gà? ..." Do đó, cùng lúc, mô hình logistic regression cần giải quyết nhiều bài toán.

Thay vì phép biến đổi tuyến tính WX cho đầu ra là một giá trị và ta dùng giá trị đó làm đầu vào cho hàm sigmoid, trong trường hợp này, mô hình logistic regression thực hiện phép biến đổi tuyến tính WX cho đầu ra là một vector và ta áp dụng hàm sigmoid lên từng phần tử của vector.

Đến đây, ta thu được một dãy các giá trị xác suất mà mỗi giá trị lần lượt tương ứng với lời dự đoán của mô hình logistic regression trên từng câu hỏi "có hình ảnh của chó hay không có hình ảnh của chó?, có hình ảnh của mèo hay không có hình ảnh của mèo? có hình ảnh của gà hay không có hình ảnh của gà? ..."

Việc tính giá trị loss của hàm binary cross entropy trong bài toán multi-label classification không quá khác so với trong bài toán binary classification. Sau khi tính loss với từng lời dự đoán của mô hình trên từng câu hỏi, thông thường, ta sẽ lấy tổng hoặc lấy trung bình các giá trị loss này để thu được giá trị loss cuối cùng.

Cụ thể, với bài toán multi-label classification gồm K lớp

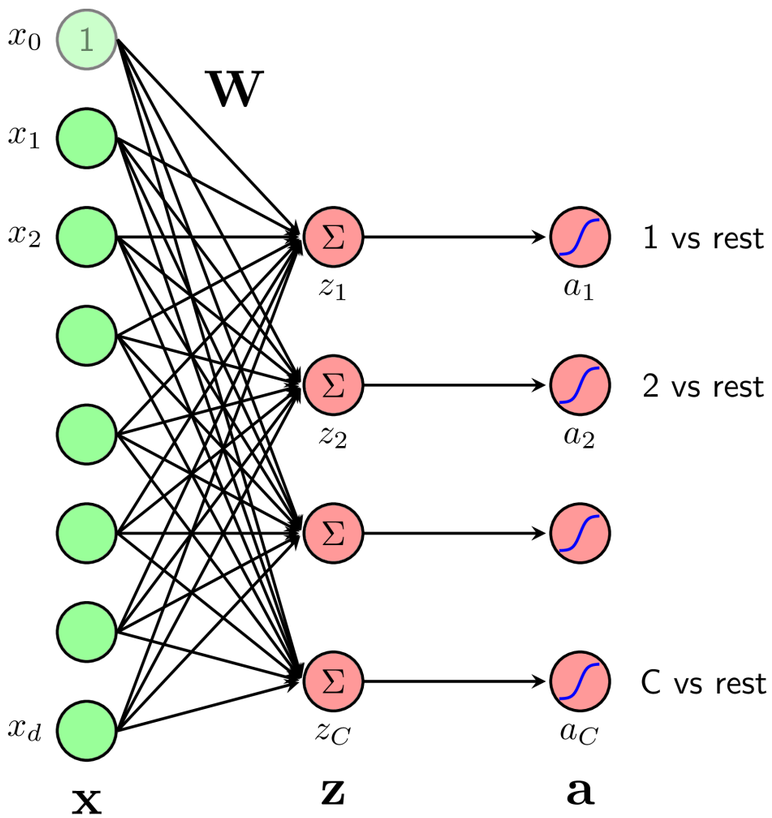
$$\mathcal{L}(\hat{y}) = \frac{1}{k}\sum_{k}^{K}\mathcal{L}(\hat{y}_k)$$

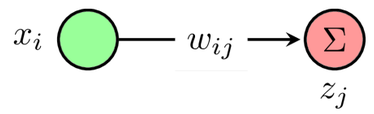
Từ đó việc tính đạo hàm cũng sẽ khác một chút

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{k}\sum_{k}^{K}\frac{\partial \mathcal{L}}{\partial \sigma_k}\cdot\frac{\partial \sigma_k}{\partial z_k}\cdot\frac{\partial z_k}{\partial W_k}$$

Đến đây, ta vẫn áp dụng thuật toán gradient descent để tối ưu W như bình thường.

Trong một số tài liệu, ta có thể gọi mô hình logistic regression giải quyết bài toán multi-label classification là mô hình logistic regression one-vs-rest.





w_{0j}: biases, don't forget!

d: data dimension

C: number of classes

$\mathbf{x} \in \mathbb{R}^{d+1}$

$\mathbf{W} \in \mathbb{R}^{(d+1)\times C}$

$z_i = \mathbf{w}_i^T\mathbf{x}$

$\mathbf{z} = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^C$

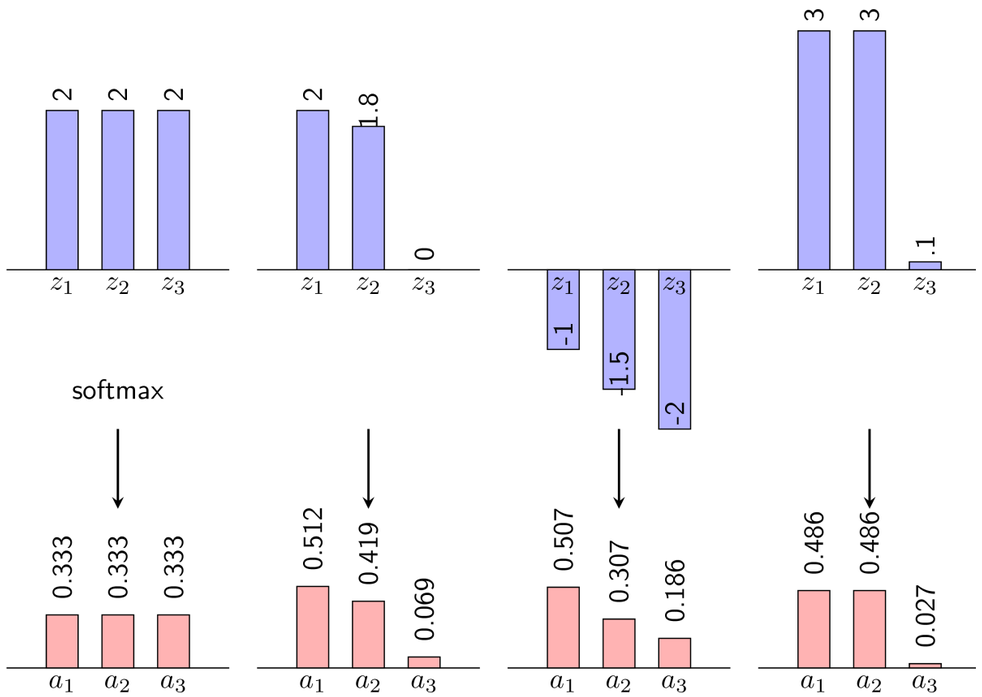
$a_i = \text{sigmoid}(z_i) \in \mathbb{R}$

$0 < a_i < 1$

5. Logistic regression giải quyết bài toán multi-class classification

Có một trường hợp giống với multi-label classification là ta làm việc với nhiều lớp dữ liệu khác nhau, nhưng khác so với bài toán multi-label classification là mỗi ***dữ liệu đầu vào chỉ có thể có một label***. Lúc này, ta đang xử lý bài toán multi-class classification.

Mô hình logistic regression giải quyết bài toán multi-class classification có thể được gọi với tên gọi khác là Softmax Regression vì ta sẽ sử dụng hàm Softmax thay thế cho hàm Sigmoid ở vị trí của một logistic activation function.



Với việc thay đổi logistic activation function, ta cần một hàm loss khác, và đó là hàm loss Categorical Cross Entropy. Khác với Binary cross entropy, hàm loss Categorical Cross Entropy tính toán giá trị loss trên tất cả các lớp trong bộ dữ liệu. Ta cũng có thể hiểu, Categorical Cross Entropy là một phiên bản khái quát hơn của Binary cross entropy.

Giả sử, ta có K lớp trong bộ dữ liệu,

$$\mathcal{L}(W) = -\sum_{k}^{K} \log \hat{y_k}^{y_k} = -\sum_{k}^{K} y_k \log \hat{y_k}$$

trong đó:

$$\hat{y_k} = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}$$

$$z_k = W_k X$$

Đến đây, cũng áp dụng chain rule, ta có

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial softmax} \cdot \frac{\partial softmax}{\partial z} \cdot \frac{\partial z}{\partial W}$$

Đạo hàm của hàm loss categorical cross entropy được tính như sau

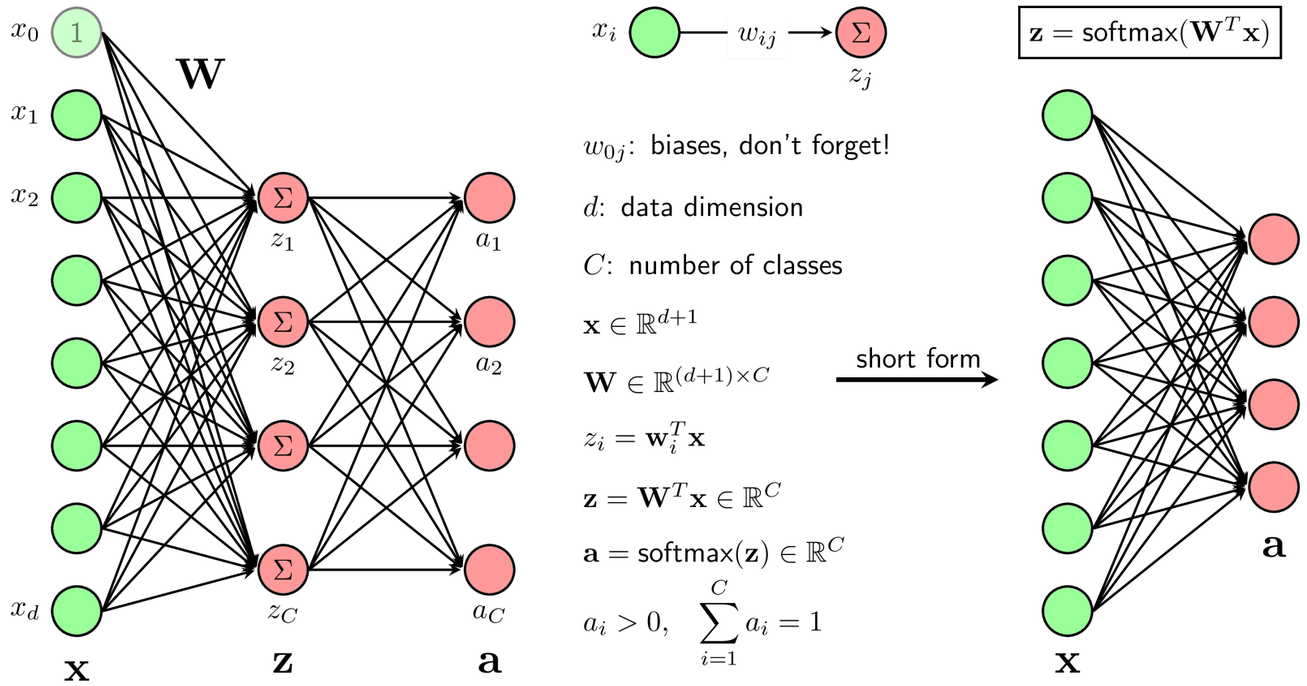
$$\frac{\partial \mathcal{L}}{\partial softmax} = \sum_k^K \frac{\partial \mathcal{L}}{\partial \hat{y}_k} = \sum_k^K \frac{y_k}{\hat{y}_k}$$

Vì softmax là hàm số nhận đầu vào là một vector, do đó, đạo hàm của hàm softmax được tính như sau

$$\frac{\partial softmax}{\partial z} = \left[\frac{\partial softmax}{\partial z_1}, \frac{\partial softmax}{\partial z_2}, \ldots, \frac{\partial softmax}{\partial z_K}\right]$$

Đạo hàm của phép biến đổi tuyến tính tương tự như trên.

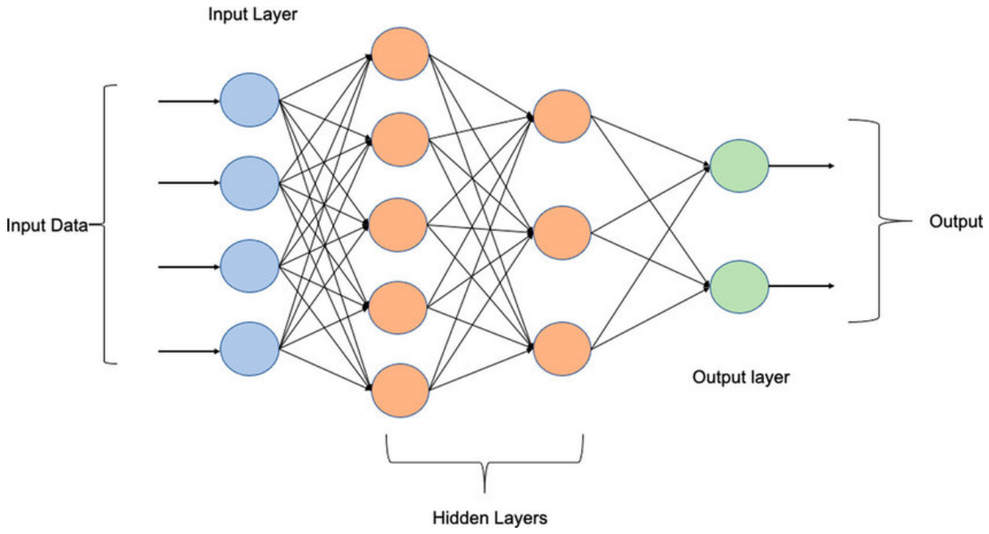
Đến đây, ta vẫn áp dụng thuật toán gradient descent để tối ưu W như bình thường.



6. Multi-layer perceptron (MLP) - Artificial Neural Network (ANN)

Đối với những bộ dữ liệu đơn giản, mô hình như Logistic regression có thể giải quyết được. Tuy nhiên, khi gặp những bộ dữ liệu phức tạp hơn và nhiều dữ liệu hơn, nếu chỉ sử dụng mô hình nhỏ như Logistic regression, ta sẽ gặp phải hiện tượng underfit.

Do đó, có một cách để giúp tăng độ phức tạp của mô hình, qua đó giúp giải quyết tốt hơn những bộ dữ liệu phức tạp, đó là xếp chồng nhiều các phép biến đổi tuyến tính lên nhau, tạo thành các lớp của mô hình. Từ đó, ta sẽ thu được một kiến trúc mô hình mới, có độ phức tạp cao hơn, được gọi là Multi-layer perceptron (MLP) hoặc mạng nơ ron nhân tạo Artificial Neural Network (ANN). Kiến trúc MLP này là nền tảng cho rất nhiều các mô hình deep learning nổi tiếng hiện nay.



Tuy nhiên, nếu ta chỉ đơn thuần xếp chồng các lớp biến đổi tuyến tính lên nhau, theo phân tích toán học, bản chất của mô hình mạng nơ ron lúc này không khác so với mô hình logistic regression đơn giản, hay nói cách khác, độ phức tạp của mô hình lúc này không được tăng lên.

Do đó, xen giữa các lớp biến đổi tuyến tính, ta cần bổ sung thêm các lớp biến đổi phi tuyến bằng việc sử dụng các activation function như ReLU, Leaky ReLU ...

Ngoài ra, với việc tăng độ phức tạp của mô hình, quá trình tối ưu mô hình hay cực tiểu hàm loss cũng trở nên khó khăn hơn, trong mạng nơ ron, ta có thể sử dụng thêm một số lớp normalization như Batch Normalization, Layer Normalization giúp ổn định hoá quá trình tối ưu của mô hình.