Neural Network And Its Fundamentals

Ranjan Mondal

JPMorgan Chase & Co
Bangalore
India

July 15, 2024

Neural Network And Its Fundamentals

Ranjan Mondal

JPMorgan Chase & Co
Bangalore
India

July 15, 2024

Applications

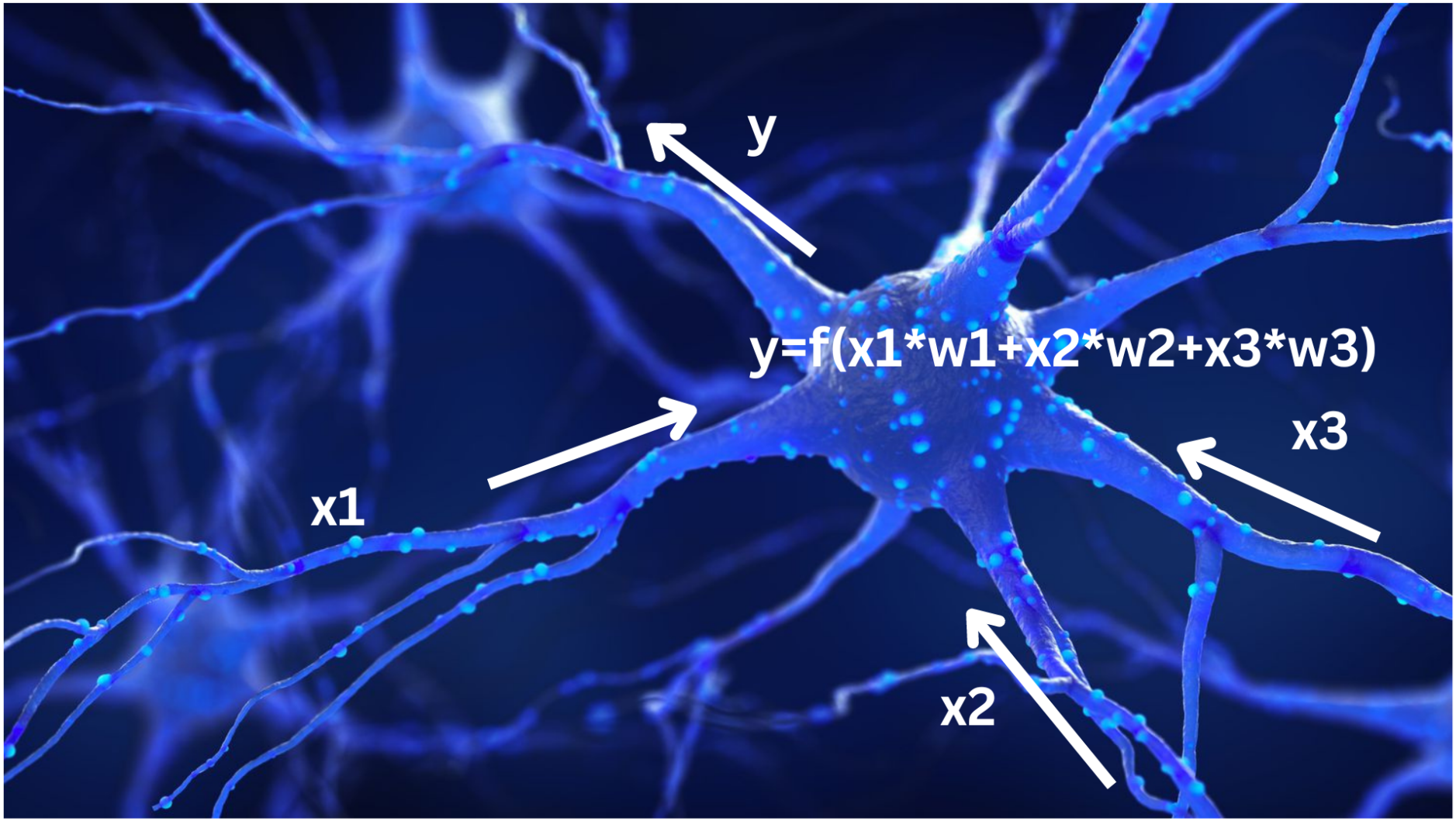
1. Computer Vision
   - Image Classification
   - Object Detection
   - Image Generation (e.g., GANs)
2. Natural Language Processing (NLP)
   - Text Classification
   - Machine Translation
   - Sentiment Analysis
   - Large Language Models (LLMs)
3. Finance
   - Stock Price Prediction
   - Fraud Detection
   - Algorithmic Trading

What is Artificial Neural Network?

What is Artificial Neural Network?
– Network of Neurons

What is Artificial Neuron ?

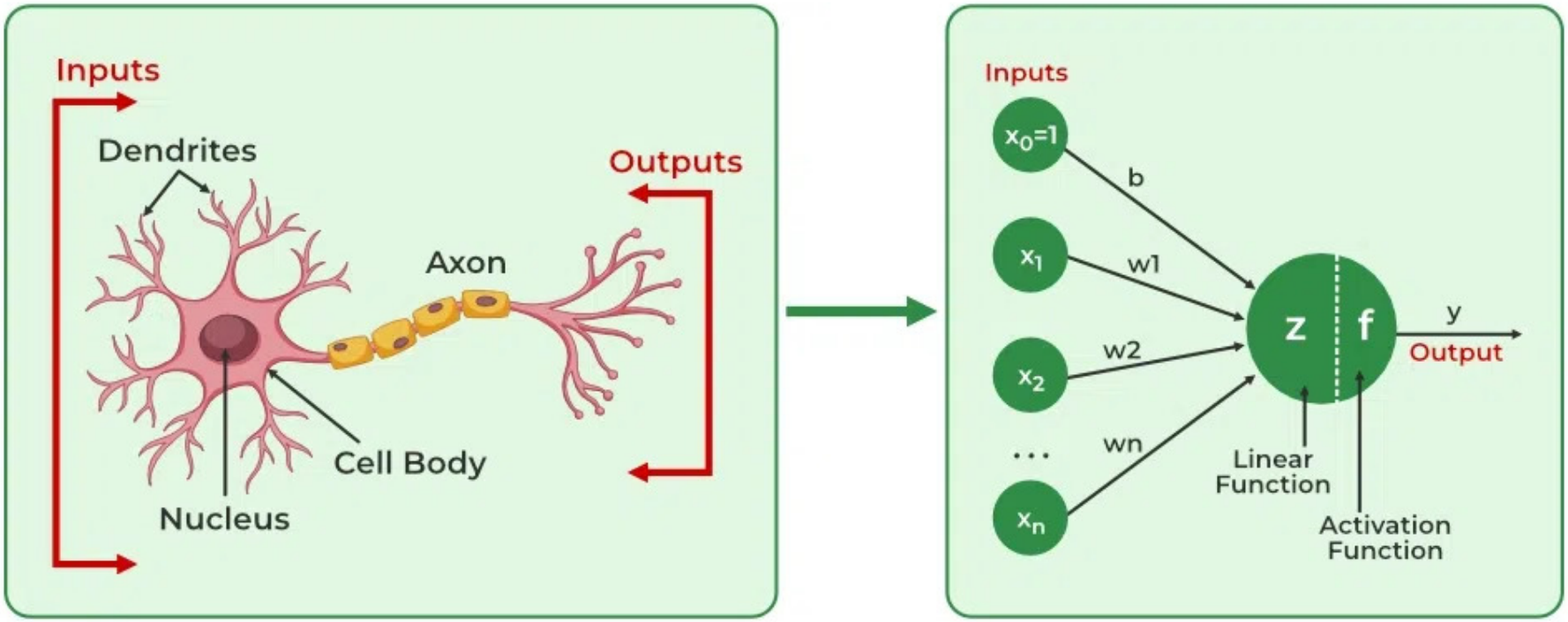
What is Artificial Neuron ?



Artificial Neuron

- Linear Combination of Input x_i with some w_i followed by a non-linear Operation

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$



Non-Linear Operation— Activation Function

Example of Non-liner operations

- Sigmoid :

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Tanh :

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- Relu :

$$max(0, x)$$

- Leaky-Relu:

$$max(\alpha x, x)$$

Neurons—parameters

Number of Parameters in a single neurons?

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2}$$

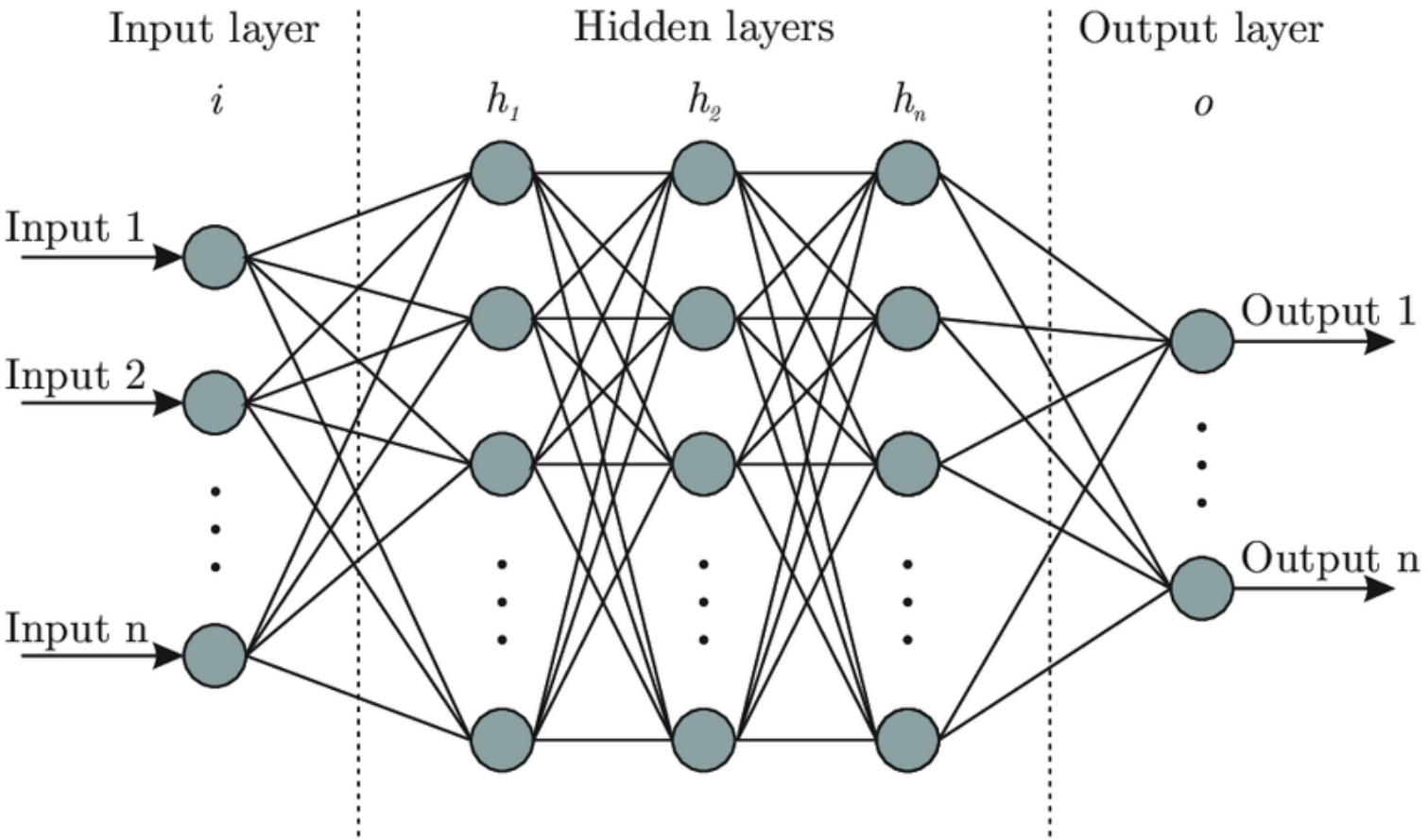
Neurons—parameters

Number of Parameters in a single neurons?

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2}$$

Ans: $n + 1$
n number of neurons and bias. There are no parameters in the activation function.

Network of Neurons

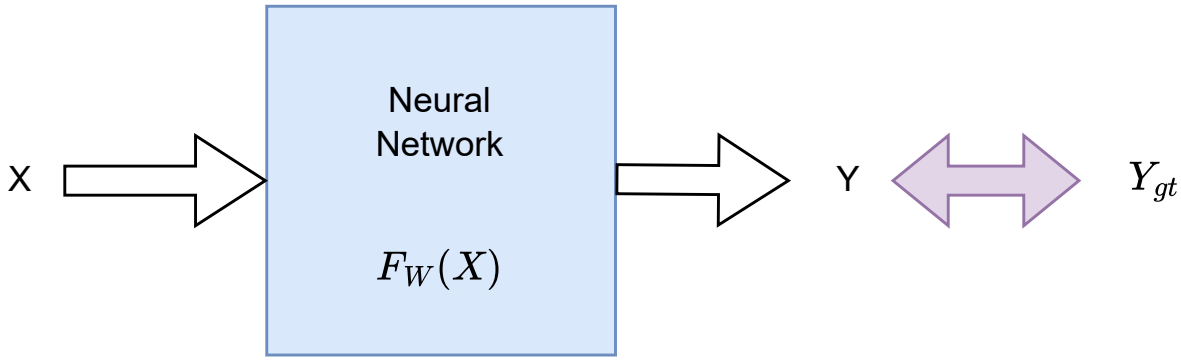


$Y = f(W_4 f(W_3 f(W_2 f(W_1 x))))$

#number of parameters: $(n+1)h_1 + (h_1+1)h_2 + (h_2+1)h_n + (h_n+1)n$

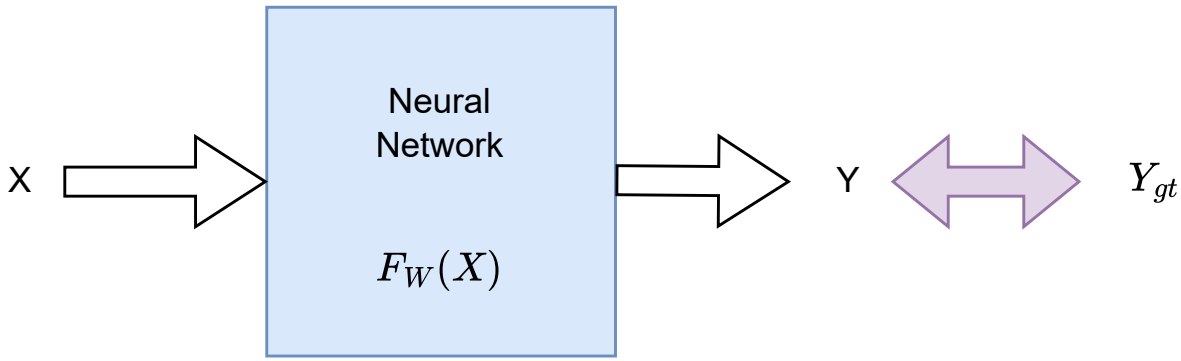
How parameters are trained?

Let $X \in R^d; Y \in R^m ; W = \{W_1, W_2, W_3 W_4\}$



How parameters are trained?

Let $X \in R^d; Y \in R^m ; W = \{W_1, W_2, W_3 W_4\}$



$$Loss = \|Y - Y_g t\| = \|F_w(X) - Y_g t\|$$

How to minimize a function?

$f(w) = w^2 - 4w + 4$
For which value w, $f(w)$ is minimum?

How to minimize a function?

$f(w) = w^2 - 4w + 4$
For which value w, $f(w)$ is minimum?

For which value of w_1, w_2, $f(w_1, w_2)$ is minimum?
$f(w_1, w_2) = (w_1 - 3)^2 + (w_2 - 2)^2$

How to minimize a function?

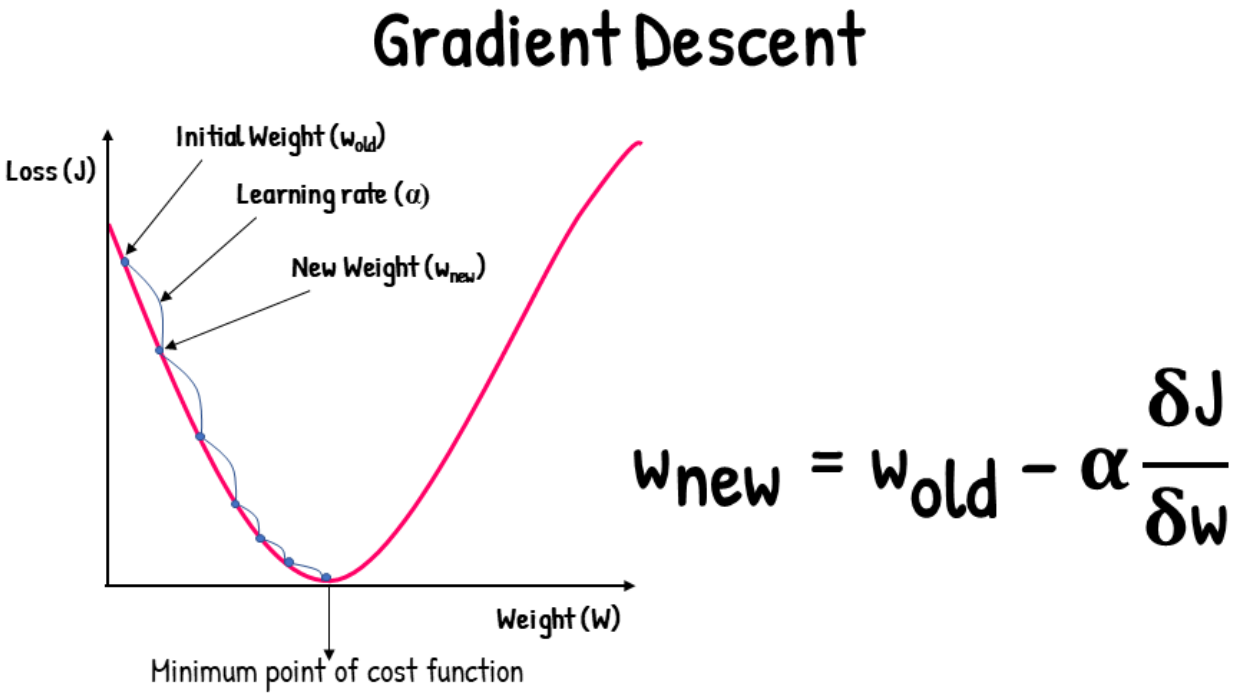
$f(w) = w^2 - 4w + 4$
For which value w, $f(w)$ is minimum?

For which value of w_1, w_2, $f(w_1, w_2)$ is minimum?
$f(w_1, w_2) = (w_1 - 3)^2 + (w_2 - 2)^2$

Other Way:
–Take the first derivative, which equals zero, compute the Hessian matrix, check determinant, solve equations.
–Gradient Descent
–Newton Raphson

Gradient Descent

$$f(w_1, w_2) = (w_1 - 3)^2 + (w_2 - 2)^2$$



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

How to minimize a loss ?

Let $L = loss(W) = \|f(W_4 f(W_3 f(W_2 f(W_1 x)) - Y_{gt}\|$
where

$$f(x) = \frac{1}{1 + e^{-x}}$$

– We want to find W that minimize L
– Calculate the gradient with respect to W and update $W = W - \alpha * \frac{\delta L}{\delta W}$

How to minimize a loss ?

Let $L = loss(W) = \|f(W_4 f(W_3 f(W_2 f(W_1 x)) - Y_{gt}\|$
– We want to find W that minimize L
Computing Gradient:

$$\frac{\delta L}{\delta W_3} = \frac{\delta L}{\delta W_4}\frac{\delta W_4}{\delta W_3}$$

$$\frac{\delta L}{\delta W_1} = \frac{\delta L}{\delta W_4}\frac{\delta W_4}{\delta W_3}\frac{\delta W_3}{\delta W_2}\frac{\delta W_2}{\delta W_1}$$

Update:

$$W_1 = W_1 - \alpha * \frac{\delta L}{\delta W_1}$$
$$W_2 = W_2 - \alpha * \frac{\delta L}{\delta W_2}$$
$$W_3 = W_3 - \alpha * \frac{\delta L}{\delta W_3}$$
$$W_4 = W_4 - \alpha * \frac{\delta L}{\delta W_4}$$

Training of Neural Network

Forward pass:

- Provide the input
- Compute output by $\phi(W_3\phi(W_2\phi(W_1X)))$

Backward Pass:

- Compute the loss
- Compute the gradient with respect to parameters
- Update the parameters

Question?

Questions?

Question?

Questions? Examples

Questions

- What are the hyperparameters?
- Is the loss function convex with respect to the parameters?
- How many layers should we use?
- What if we don't use any activation function?
- What activation function should we use?
- What learning rate should we use?
- What loss function should we use?
- What optimizer should we use ?
- What function should we use at last layer ?

Why Does Neural Network Work?

Universal Approximation Theorem

- Neural network with a single hidden layer containing a finite number of neurons can approximate any continuous functions.
- i.e $W_2\phi(W_1X)$ Can approximate any continuous function

Any other Function that represents any continuous function ?

Why Does Neural Network Work?

Universal Approximation Theorem

- Neural network with a single hidden layer containing a finite number of neurons can approximate any continuous functions.
- i.e $W_2\phi(W_1X)$ Can approximate any continuous function

Any other Function that represents any continuous function ?

- Taylor Series
- Fourier Series
- Wavelet Transform
- Piecewise Linear Function s
- Polynomial Functions

Why Does Neural Network Work?

Other Function that represents any continuous function?

- Taylor Series

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{3}$$

- Fourier Series
- Wavelet Transform
- Piecewise Linear Functions

$$f(x) = \begin{cases} m_1 x + b_1 & \text{if } x < c_1 \\ m_2 x + b_2 & \text{if } c_1 \leq x < c_2 \\ \vdots & \vdots \\ m_n x + b_n & \text{if } x \geq c_{n-1} \end{cases} \tag{4}$$

- Polynomial Functions

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \tag{5}$$

Why Does Neural Network Work?

Example

Optimization — Gradient Descent

Vanilla SGD

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

Momentum

$$w_{t+1} = w_t - \alpha m_t$$

$$m_t = \beta m_{t-1} + (1 - \beta) \frac{\partial L}{\partial w_t}$$

Adagrad

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$v_t = v_{t-1} + \left[\frac{\partial L}{\partial w_t} \right]^2$$

Optimization — Gradient Descent

RMSprop

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$v_t = \beta v_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t} \right]^2$$

Adadelta

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \epsilon}}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$D_t = \beta D_{t-1} + (1 - \beta)[\Delta w_t]^2$$

$$v_t = \beta v_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t} \right]^2$$

Optimization — Gradient Descent

Adam

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

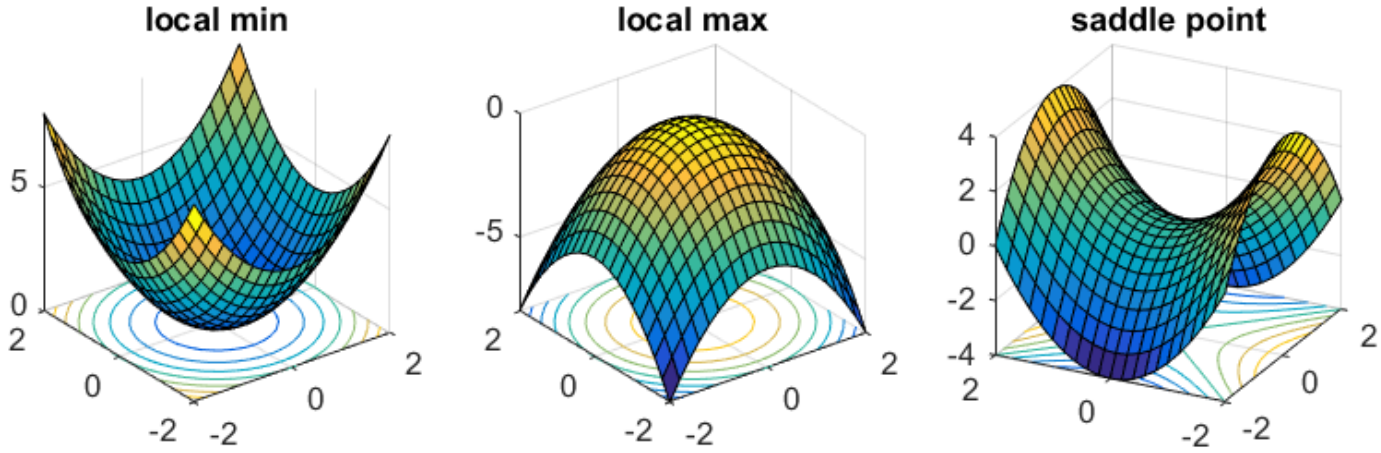
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t} \right]^2$$

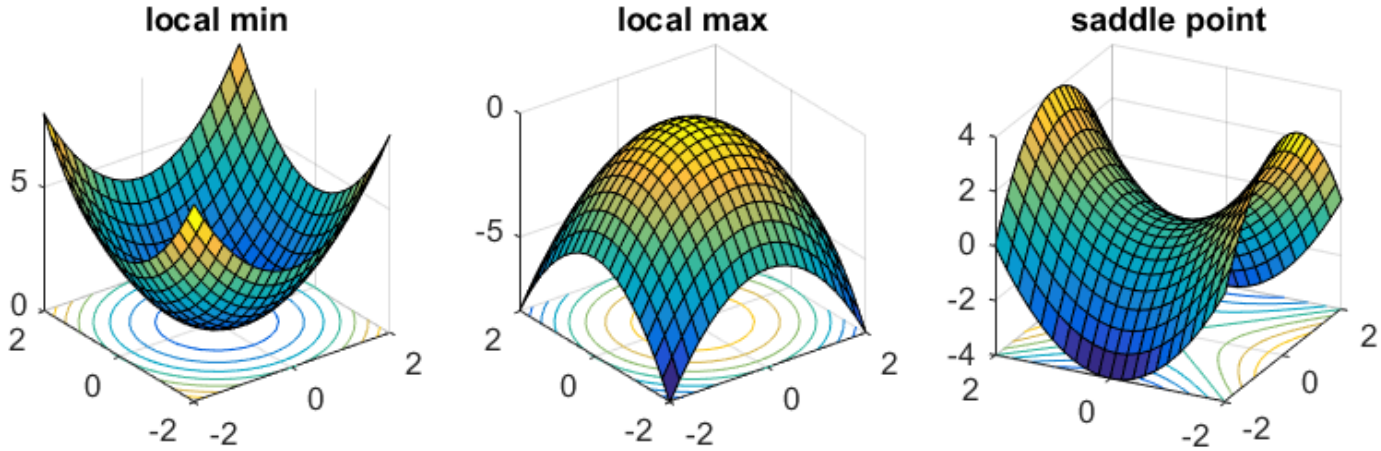
Optimization of NN parameters

Loss surface with respect to two NN parameters



Optimization of NN parameters

Loss surface with respect to two NN parameters



In real, there will be no two parameters. There will be billions of parameters

Optimization of NN parameters

Let $W \in \mathbb{R}^n$ be the parameters of the neural network. The gradient with respect to the loss L is:

$$\nabla L(W) = \begin{pmatrix} \frac{\partial L}{\partial W_1} \\ \frac{\partial L}{\partial W_2} \\ \vdots \\ \frac{\partial L}{\partial W_n} \end{pmatrix}$$

The Hessian matrix H is given by:

$$H = \begin{pmatrix} \frac{\partial^2 L}{\partial W_1^2} & \frac{\partial^2 L}{\partial W_1 \partial W_2} & \cdots & \frac{\partial^2 L}{\partial W_1 \partial W_n} \\ \frac{\partial^2 L}{\partial W_2 \partial W_1} & \frac{\partial^2 L}{\partial W_2^2} & \cdots & \frac{\partial^2 L}{\partial W_2 \partial W_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial W_n \partial W_1} & \frac{\partial^2 L}{\partial W_n \partial W_2} & \cdots & \frac{\partial^2 L}{\partial W_n^2} \end{pmatrix}$$

If 2nd order partial derivative are continuous then H is symmetric.

Optimization of NN parameters

Hessian matrix denotes the 2nd derivative of loss with respect to the parameters
The following is what you can say about the network.After computing eigen values of the Hessian.

- Positive Eigenvalues: Corresponds to the directions in the parameter space where loss function have a convex shape(curves upward)
- Magnitude of Eigenvalues: Smaller the eigenvalues, flatter the surface

Small vs Large network

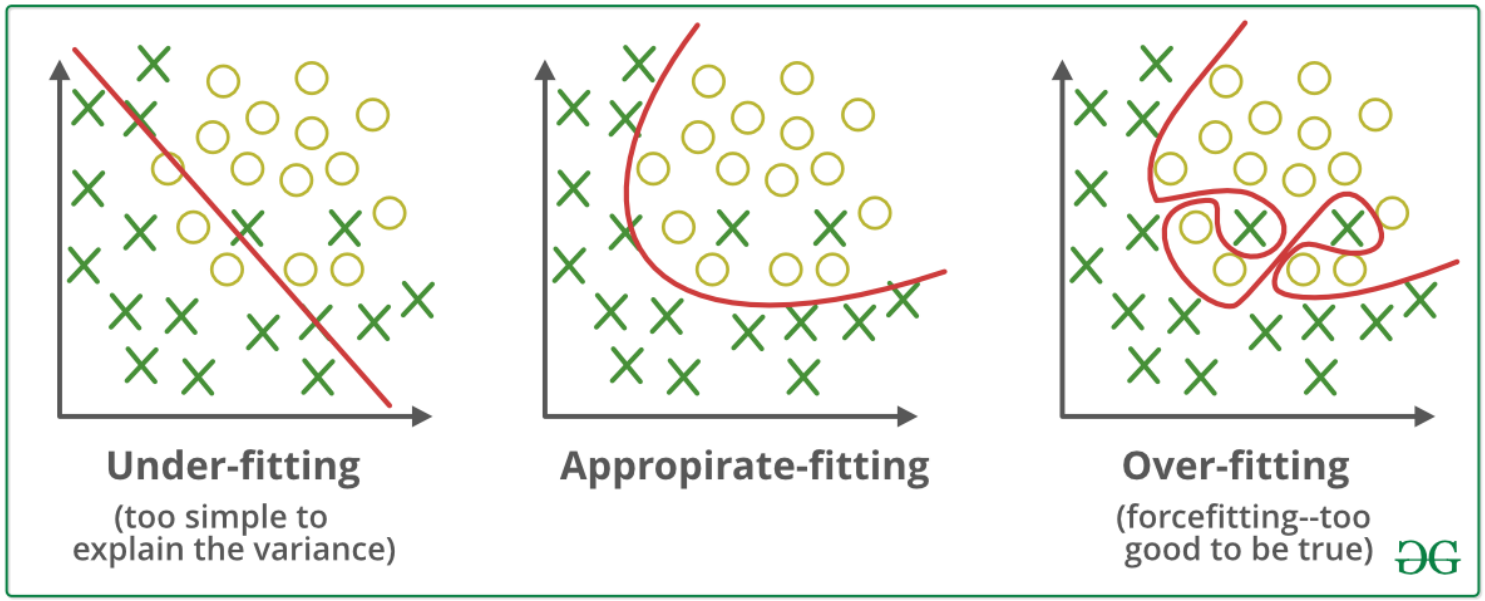
Consider Hessian matrix of small and larger network with respect to the parameters
Distribution of eigenvalues are more concentrated toward smaller value for larger network than smaller network

- Global minima is very near to local minima for larger network
- More generalization for larger network
- Larger network have higher saddle points(have escape route) and local minima then smaller network.

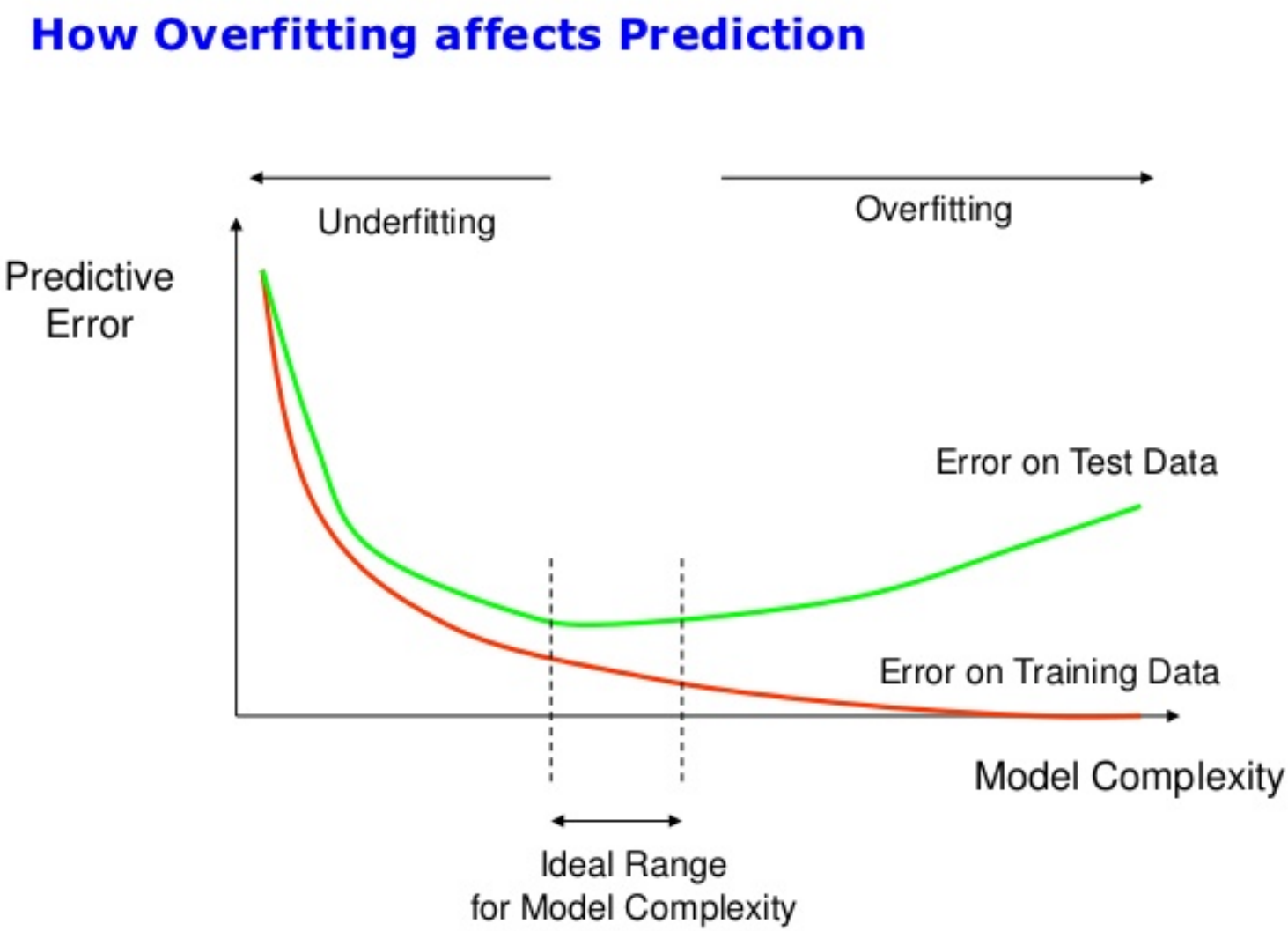
Example

Example

Overfitting



Overfitting



Overfitting

How to handle over-fitting?

- Decrease the model complexity
- Change the objective function

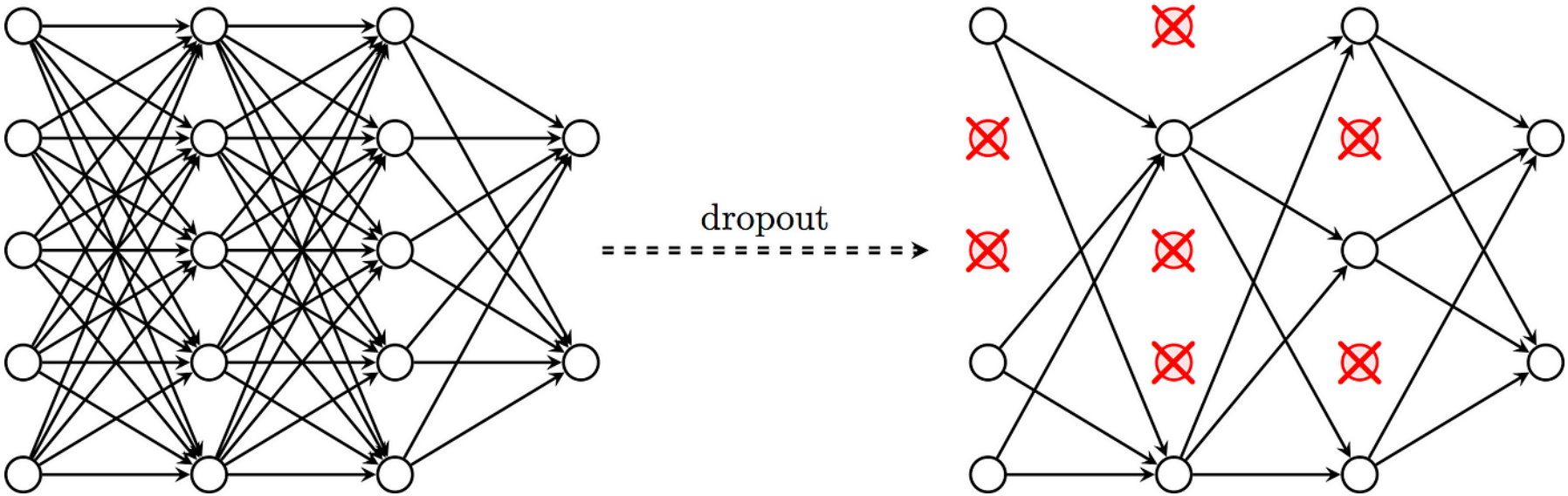
Overfitting — Change the objective function

Let L be the existing loss

- L1 Regularization: New loss$= L + \alpha \| W \|_1$
- L2 Regularization: New Loss$= L + \alpha \| W \|_2$

Q: Why do we want to minimize the weights along with original loss?

Dropout



Batch Normalization

- Batch Normalization: Normalizing input before activation within the batch.
-

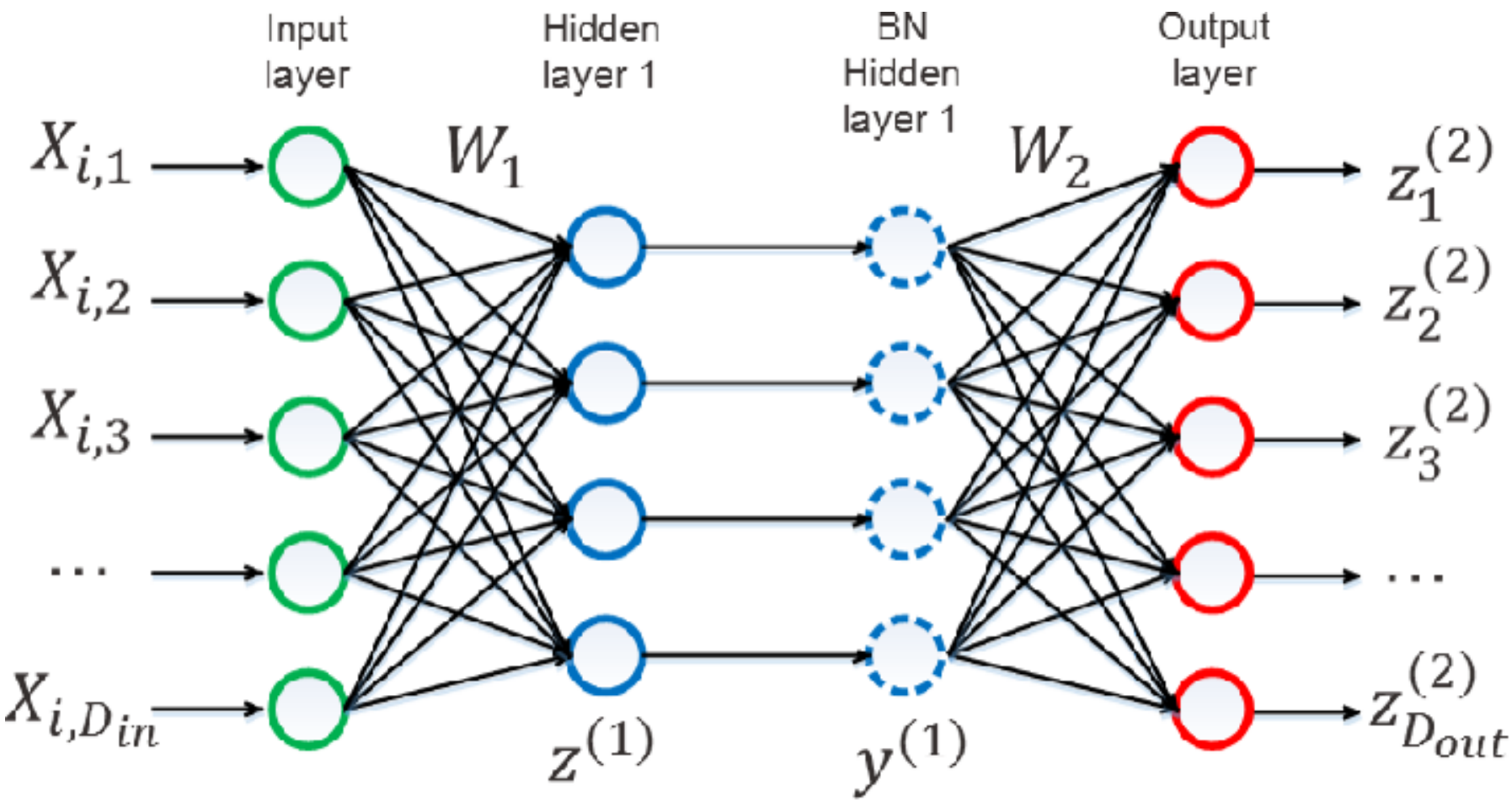
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad \text{(normalized)} \tag{6}$$

$$y_i = \gamma \hat{x}_i + \beta \quad \text{(scaled and shifted)} \tag{7}$$

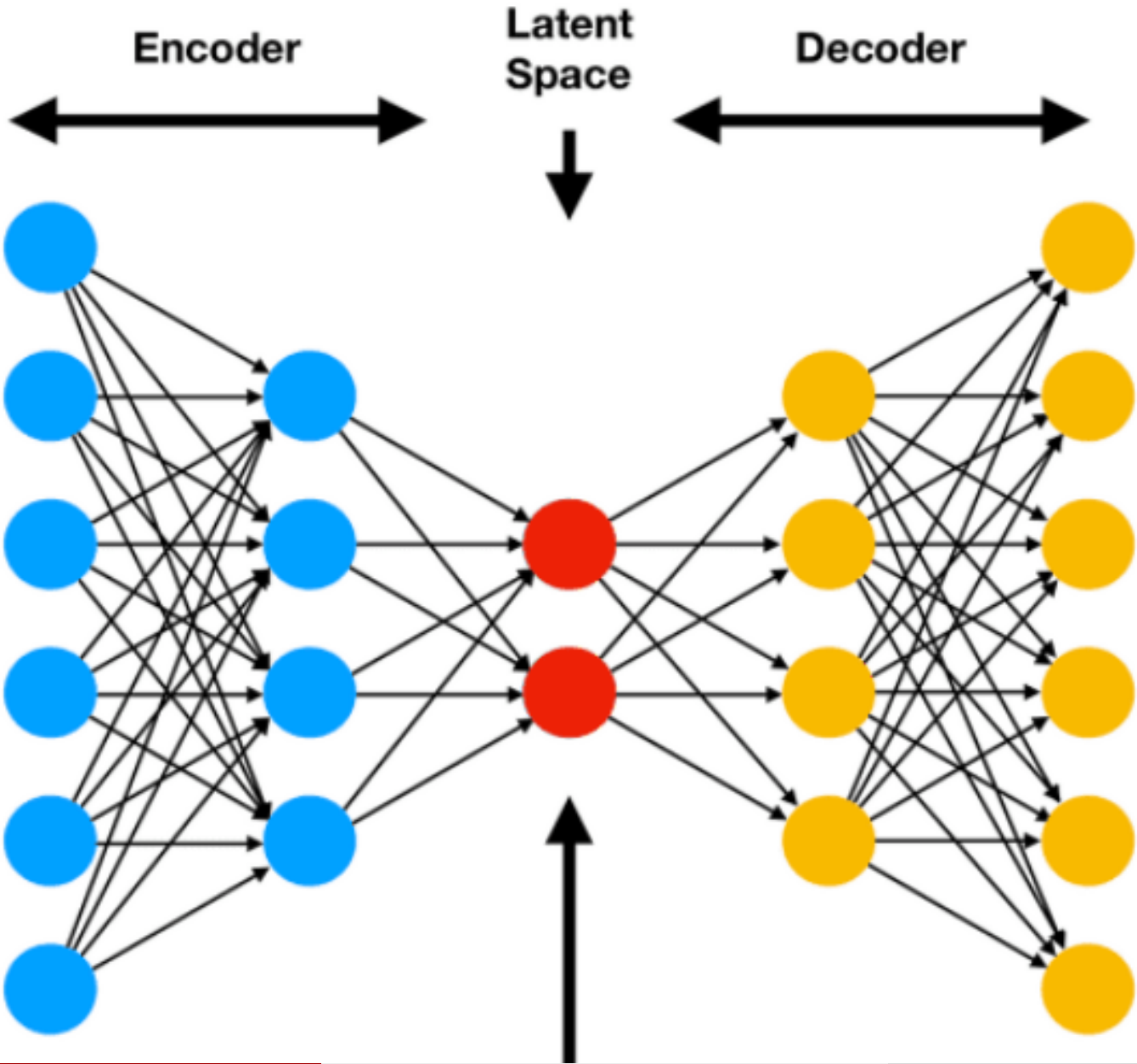
where:

  - x_i is the input.
  - μ_B is the mean of the batch.
  - σ_B^2 is the variance of the batch.
  - ϵ is a small constant added to the variance to avoid division by zero.
  - γ and β are trainable parameters (scale and shift, respectively).

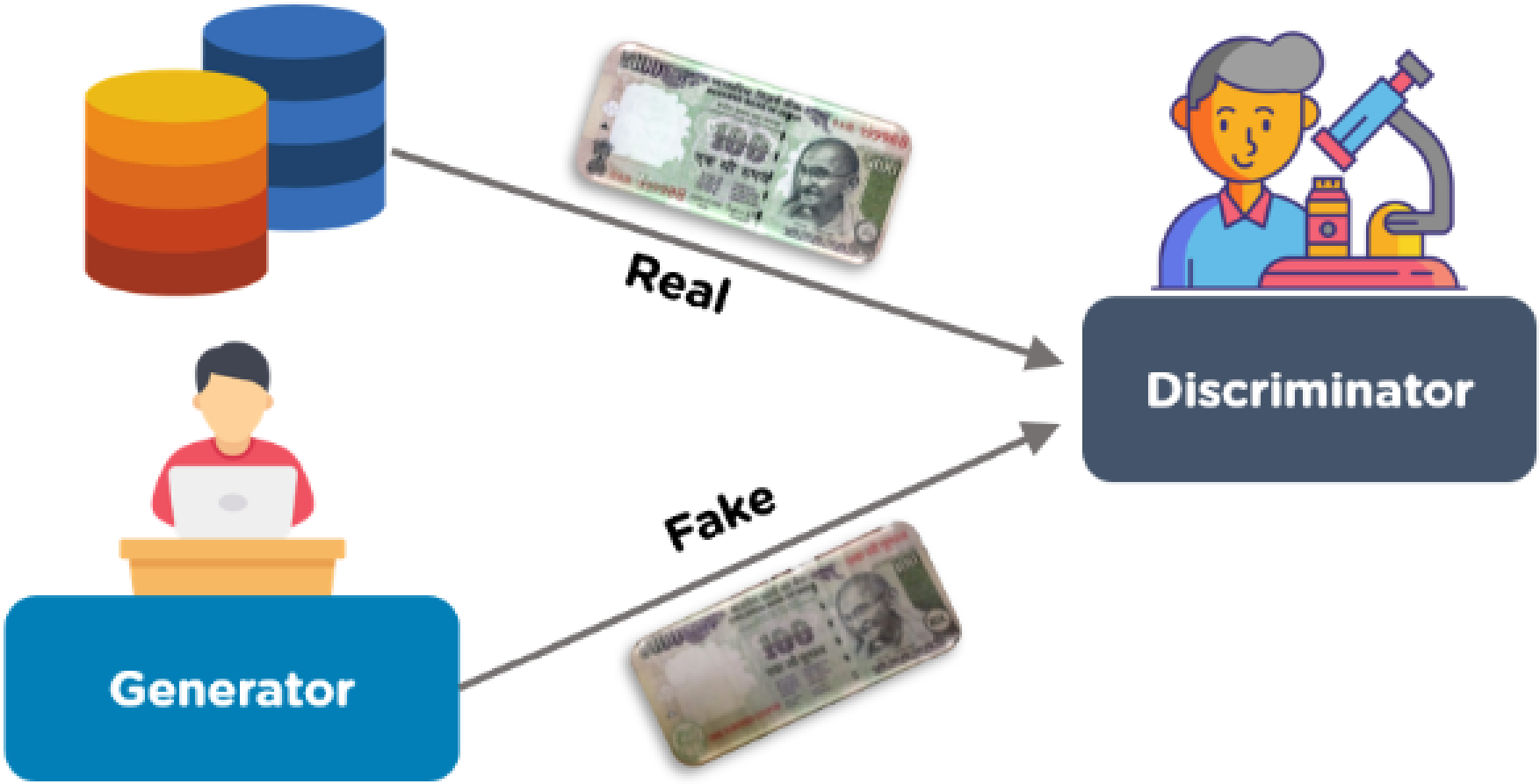
Batch Normalization



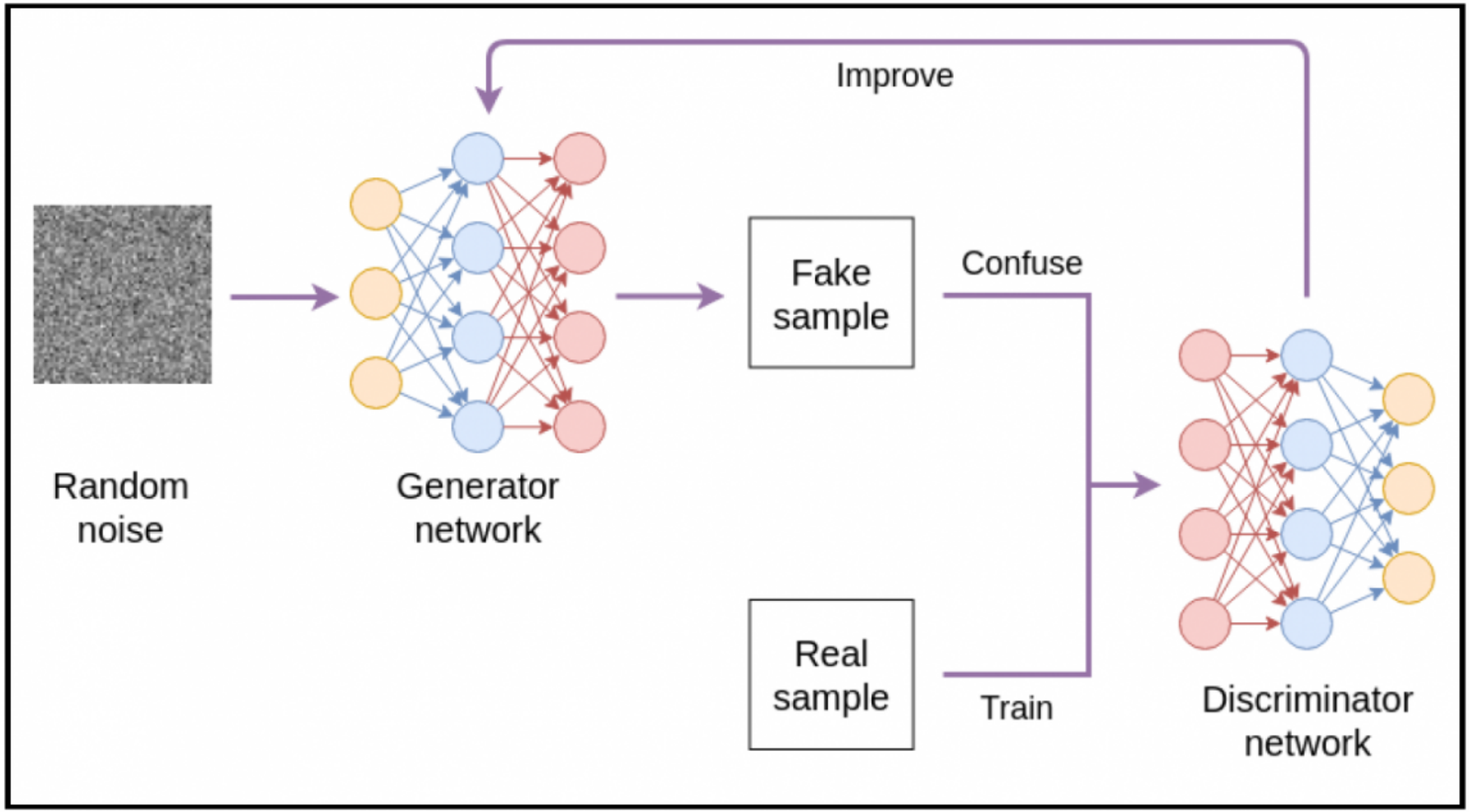
Auto Encoder



Generative Adversarial Network



Generative Adversarial Network



Generative Adversarial Network

The loss functions for GAN can be defined as follows:

Generator Loss:

$L_G = -\mathbb{E}_{z \sim p_z(z)}[\log(D(G(z)))]$

Discriminator Loss:

$L_D = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$

Questions?

Thank You