Regularized autoencoder :-

①

It include the regularization term to prevent overfitting & help model to learn better representations.
The regularization add constrain to the encoding Process forcing the model to learn meaningful features instead of memorizing the input data.

Types of regularized autoencoder :-

For. ① sparse autoencoder.
② Denoising "
③ Contractive "
④ Stochastic autoenoder.

① sparse autoencoder :- Forces autoencoder to active only few neurons at a time. making it focus on key features.

How it works :- Adds sparsity constraint to ensure that only small number of neurons in hidden layer are active.

mathematical expression:-

$$L = \text{Reconstruction Loss} + \lambda \Sigma |h(x)|$$

wher $h(x)$ is activation of hidden neurons.
λ is regularization parameter.

Contractive Autoencoder :-

(2)

* It is regulaization technique.
* The objectives of autoencoder is to have robust learned representation which is less sensitive & to small variation in data.
* Robustness of the representation for data is don by applying a penilty term to the loss function.
* contractive autoencoder is another regularization technique just like sparse & denoising autoencoder..
* The regulizer corresponds to the frobenius norm of jacobian matrix of encoder activations with respect to the i/p.
* Frobenius norm of Jacobian matrix for hidden layer is calculated with respect to input & it is basically sum of square of elements.

Regulaization Technique :- which help to prevent overfitting by adding constraints to model complexity. L1 & L2 Regularization.

The model learns that patterns which are not work well on new data.

* Contractive Autoencoder focus on model to learn only most important patterns

* It helps model become more robust & avoid overfitting.

Contractive autoencoder ensure the small change in the input donot cause big changes in the learned representation.

* Stochastic Encoder & decoder :-

In traditional autoencoder & decoder are deterministic Meaning that same for the input x, they always produce the same encoded representation $h(x)$ & reconstructed output $\hat{x}$.

Stochastic autoencoder randomness is introduced in either the encoder, decoder or both to make model more robust.

Stochastic encoder :-

A stochastic encoder add randomness to hidden representation $h(x)$ instead of producing a fixed value.

* Mathematical Representation.

$$h(x) = f(x).$$

Stochastic Decoder :- It doesnot always produce the same $\hat{x}$ for same encoded $h(x)$. It generate the random variations of $\hat{x}$.

$$\hat{x} = g(h)$$

LSTM, ... , ...

(2) Denoising Autoencoder :-

(1)

* It is special type of autoencoder which is trained to remove noise from corrupted input data.

* It takes noisy input like blurry photo & reconstruct the original clean version.

* Architecture of denoising Autoencoder :-

Noisy input :- The input data is artificially corrupted with noise.

Encoder :- The noisy input compressed into lower dimentional lentent Representation.

Decoder :- Reconstruct the original from compressed data.

Loss fu^n :- Measure the difference between (reconstructed output (Decoder) or original i/p) → Encoder.

Working of Denoising Autoencoder :

1. Corrupting input :- Noise can be added to input data before feeding it to autoencoder.

Types of Noise :-

Gaussian noise :- noise sampled from a normal distribution.

Salt and papper noise :- Randomly turing some pixels completely white or blaile.

Dropout Noise :- Randomly setting some input value to zero

2. Encoding the noisy input :- The encoder compress the noisy input into latent representation. It learns robust features that helps reconstruct the clean version of data.

3. Decoding to remove :- The decoder reconstruct the original, noise free data.

The network retrained to minimize the reconstruction error. (MSE)

* Mathematical Notation :-

where

x = original input (clean).

$\tilde{x}$ = Noisy input (corrupted version of x)

f_θ = Encoder function.

g_ϕ = Decoder function.

$\hat{x}$ = Reconstructed output.

θ - represent the encoder parameters (weights & bias)

ϕ - represent the decoder parameters. (" " ").

The

$$h = f_\theta(\tilde{x}) = \sigma(W_e \tilde{x} + b_e)$$

$$\hat{x} = g_\phi(h) = \sigma(W_d h + b_d)$$

eg. Let's set original clean data is (3)

$$x = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix}$$

Now add noise to create corrupted version $\tilde{x}$.

$$\tilde{x} := x + \epsilon$$

where ϵ - random noise.

$$\epsilon = \begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$$

Thus, noisy input become :-

$$\tilde{x} = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.3 \end{bmatrix}$$

2. Encoding Step :- Encoder transforms the noisy input $\tilde{x}$ into lower dimensional hidden representation h.

$$h = f_\theta(\tilde{x}) = W_e \tilde{x} + b_e$$

where

$$W_e = [0.5 \quad 0.4], \quad b_e = [0.1]$$

$$h = [0.5 \quad 0.4] \times \begin{bmatrix} 1.0 \\ 0.3 \end{bmatrix} + [0.1]$$

$$= [(0.5 \times 1.0) + (0.4 \times 0.3)] + [0.1]$$

$$= [0.5 + 0.12 + 0.1] = \boxed{0.72}$$

so, encoded hidden representation :- $h = 0.72$

(7)

3. Decoding step :- The decoder reconstruct the clean data x from h :

assume $W_d = \begin{bmatrix} 1.2 \\ 0.8 \end{bmatrix}$ $b_d = \begin{bmatrix} -0.1 \\ 0.05 \end{bmatrix}$

$$\hat{x} = g\phi(h) = W_d h + b_d$$

$$\hat{x} = \begin{bmatrix} 1.2 \\ 0.8 \end{bmatrix} \times 0.72 + \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}$$

$$= \begin{bmatrix} (1.2 \times 0.72) \\ (0.8 \times 0.72) \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}$$

$$= \begin{bmatrix} 0.864 \\ 0.576 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.05 \end{bmatrix}$$

The denoised output is :- $\hat{x} = \begin{bmatrix} 0.764 \\ 0.626 \end{bmatrix}$

4. Loss fun :-

$$\frac{1}{n} \sum (x - \hat{x})^2$$

$$= \frac{1}{2} \left((0.8 - 0.764)^2 + (0.4 - 0.626)^2 \right)$$

$$= \frac{1}{2} \left((0.036)^2 + (-0.226)^2 \right)$$

$$= \frac{1}{2} (0.001296 + 0.051076)$$

$$= \frac{1}{2} \times 0.052372 = 0.026186$$

The loss is 0.026186 which tell us how far our denoised output from clean data