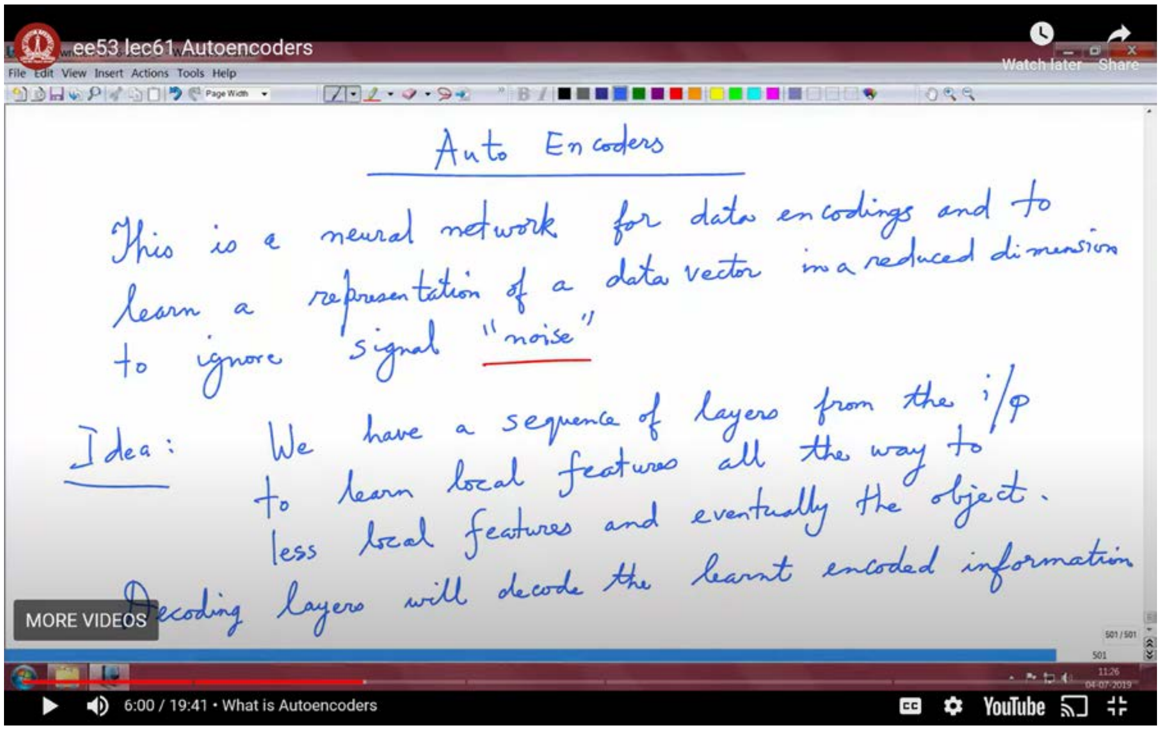
Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani
Department of Electronic System Engineering
Indian Institute of Science, Bengaluru

Lecture – 61
Autoencoders

In our discussion of advanced topics in deep learning, let's delve into autoencoders and denoising autoencoders. These concepts have fascinating applications, and their underlying ideas are quite straightforward. Let's start with the motivation behind them.

(Refer Slide Time: 06:00)



Auto Encoders

This is a neural network for data encodings and to learn a representation of a data vector in a reduced dimension to ignore signal "noise"

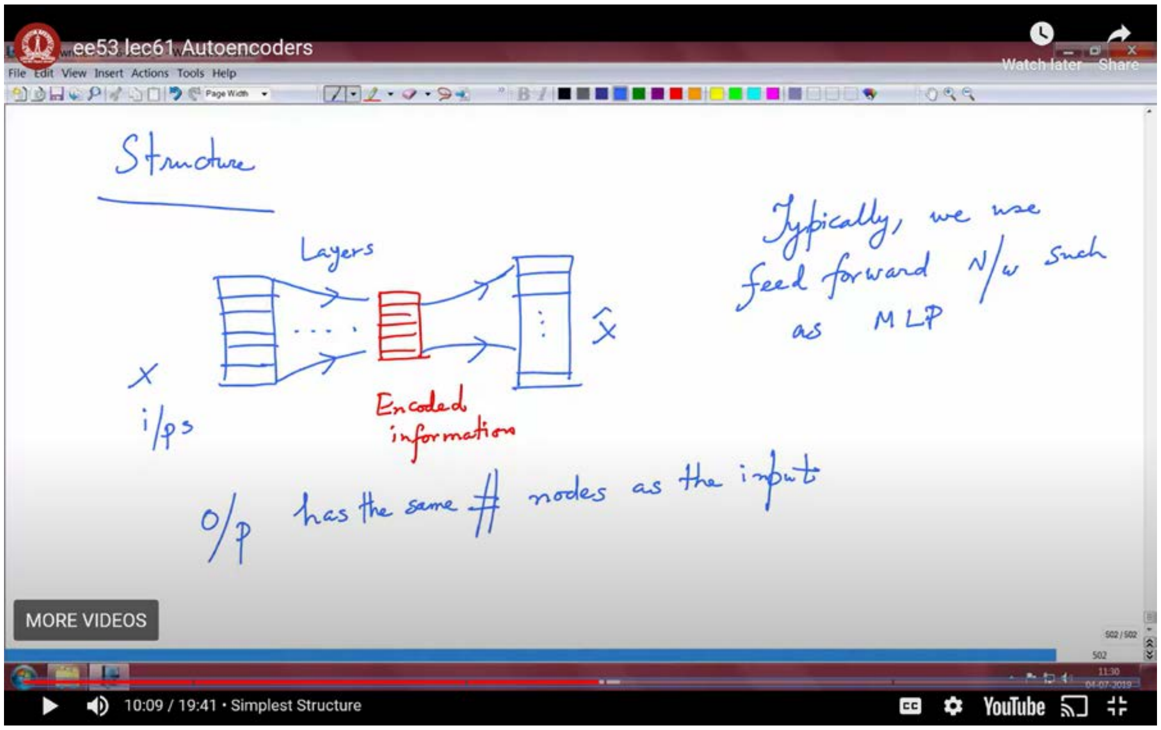
Idea: We have a sequence of layers from the i/p to learn local features all the way to less local features and eventually the object.

Decoding layers will decode the learnt encoded information

Imagine you have an object and you want different layers of a neural network to hierarchically learn various features. The goal is to establish a hierarchy of mappings so that each layer learns different aspects of the object. As we progress through this hierarchy, we aim to achieve a more compact representation of the input data. This process involves encoding the data into a lower-dimensional, more efficient representation by removing redundancy. This encoded state is a compact version of the input vectors.

Once we have this compact representation, we need to decode it back into the original form. This requires a second mapping, a non-linear transformation, to reconstruct the output from the encoded state. Essentially, this process is akin to coding: we map from a higher-dimensional space to a more compact lower-dimensional representation.

(Refer Slide Time: 10:09)



Typically, we use feed forward N/w such as MLP

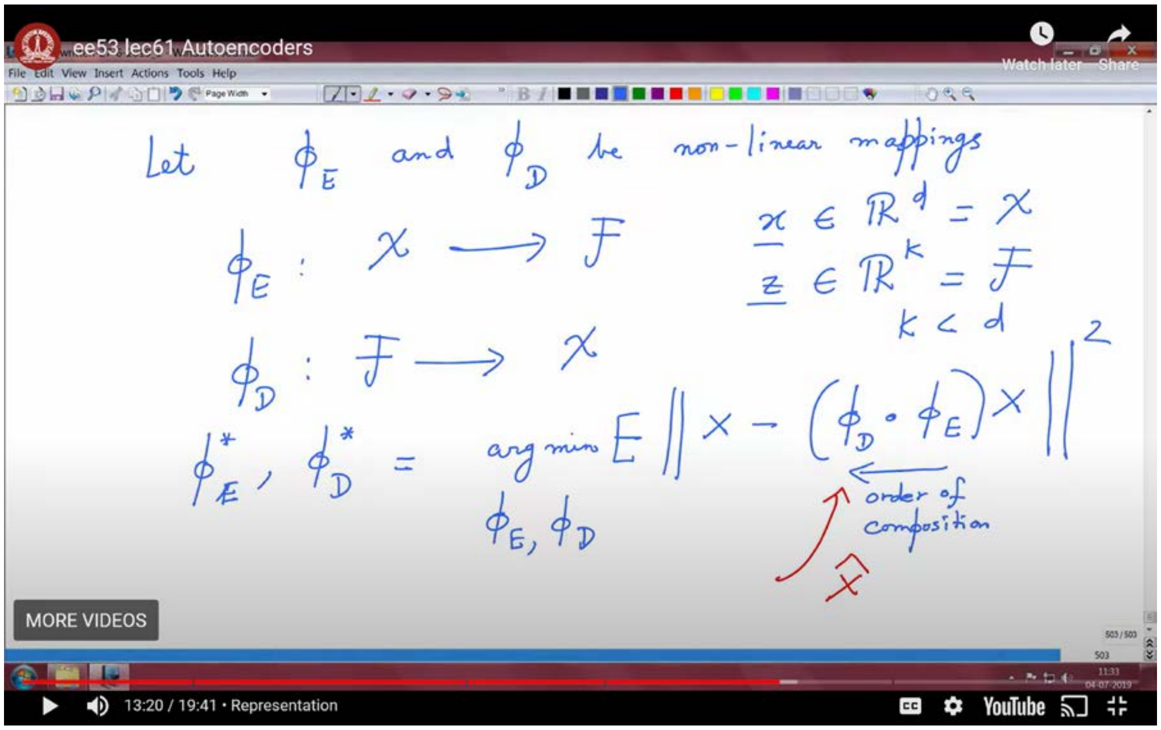
O/P has the same # nodes as the input

The core idea is to optimize these mappings so that the decoding process retains as much fidelity as possible, according to some distortion metric. This optimization involves adjusting the parameters of the neural network to achieve accurate reconstruction.

To put it simply, an autoencoder is a type of neural network designed to encode data into a lower-dimensional space and then decode it back. The objective is to learn a representation that reduces dimensionality while minimizing noise and preserving important features of the data.

The basic idea is that, through a series of layers, the network learns local features and progressively more abstract features. For example, when learning to recognize an image of a cat, the initial layers might focus on detecting basic features like the cat’s ears and eyes. As you move deeper, the network captures more complex details such as the cat's whiskers and other subtle features. Eventually, the deeper layers of the network assemble these local features into a comprehensive understanding of the object.

(Refer Slide Time: 13:20)


ee53 lec61 Autoencoders
File Edit View Insert Actions Tools Help
Watch later
Share
MORE VIDEOS
13:20 / 19:41 • Representation
YouTube

Let ϕ_E and ϕ_D be non-linear mappings

$\phi_E : \mathcal{X} \longrightarrow \mathcal{F}$ $\quad \underline{x} \in \mathbb{R}^d = \mathcal{X}$

$\underline{z} \in \mathbb{R}^k = \mathcal{F}$

$k < d$

$\phi_D : \mathcal{F} \longrightarrow \mathcal{X}$

$$\phi_E^*, \phi_D^* = \underset{\phi_E, \phi_D}{\arg\min}\, E \left\| x - (\phi_D \circ \phi_E) x \right\|^2$$

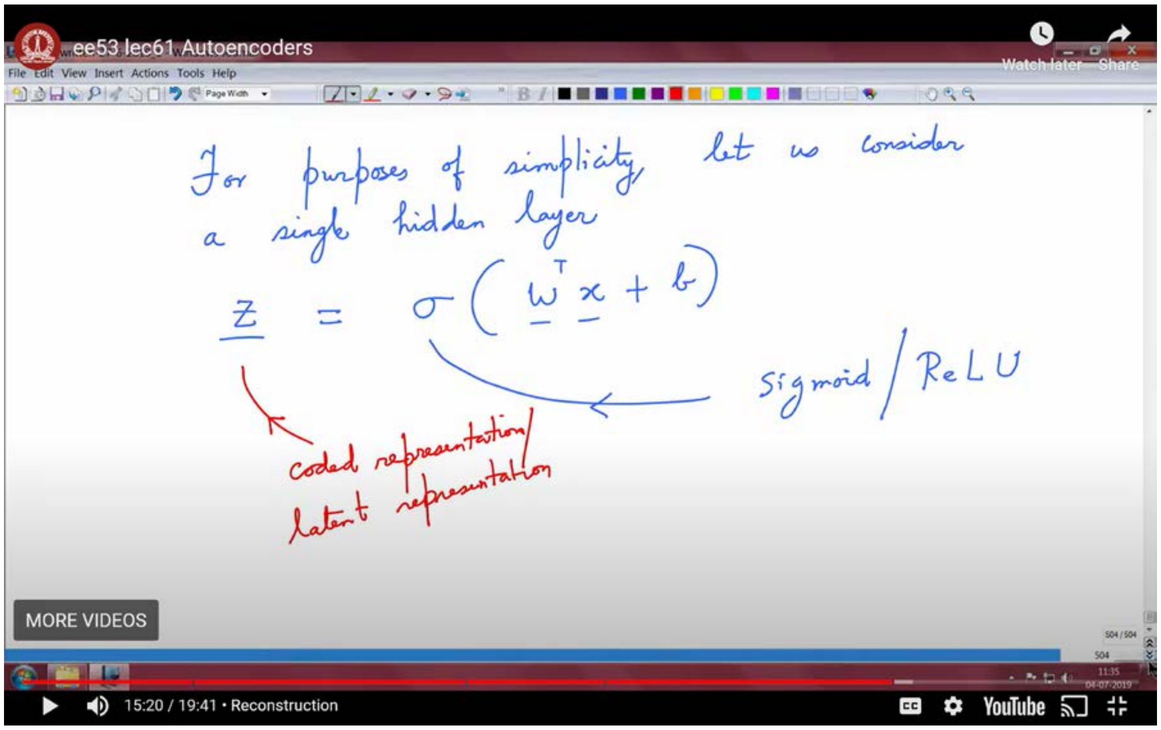
order of composition

$\hat{x}$

Thus, autoencoders use this hierarchical learning approach to encode and decode data, optimizing the representation to effectively capture and reconstruct the essential features of the input.

In a sense, there is a concept of multi-resolution at play when learning about an object, which aligns with practical observation. For instance, when I look at an image of a scene, my initial perception captures the dominant features. Imagine a beautiful portrait of a landscape with mountains and a river. At first glance, I recognize it as a landscape and identify prominent elements like the mountains and the river. This initial level of approximation provides a broad understanding of the scene.

(Refer Slide Time: 15:20)



For purposes of simplicity let us consider a single hidden layer

$$\underline{z} = \sigma(\underline{w}^T \underline{x} + b)$$

Sigmoid / ReLU

coded representation / latent representation

As I delve deeper, I notice additional details such as the depth of the valleys, the peaks, and the number of peaks, these are more refined and less immediately obvious features. Similarly, neural networks use this hierarchical approach to learning by processing features through successive layers.

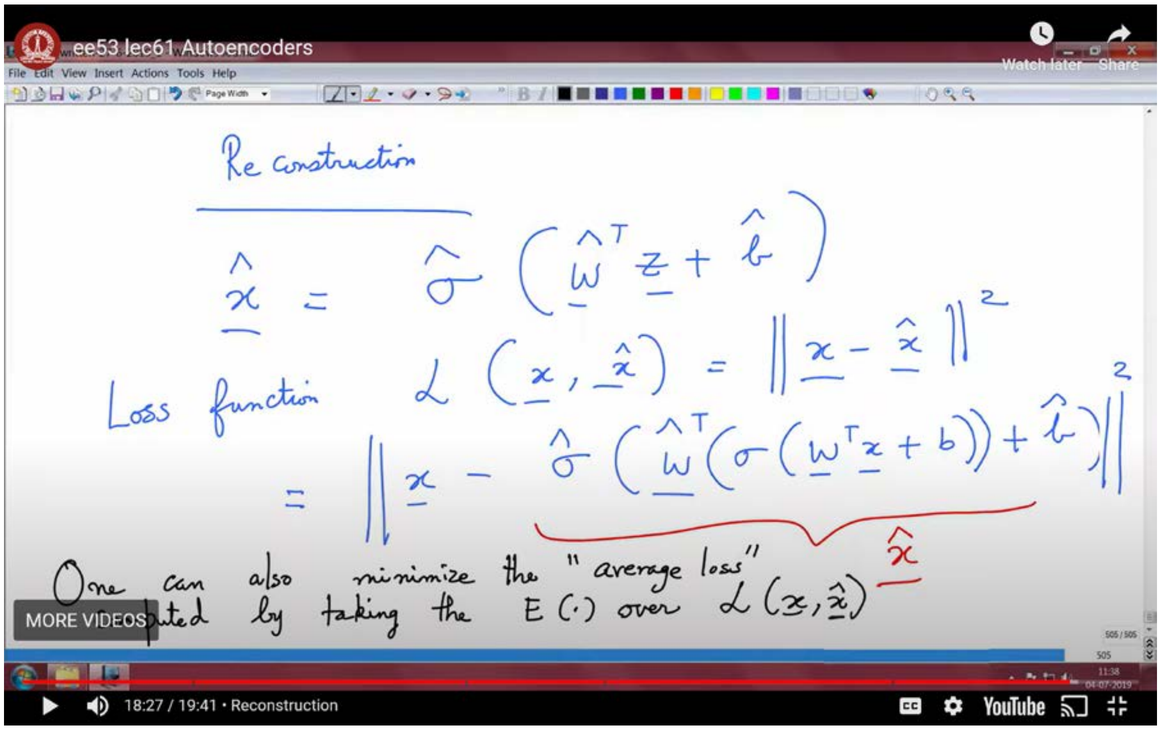
To illustrate the simplest structure, consider the following: I have an input x and a series of layers represented by bold arrows in the network. The first stage is encoding, depicted here with a different color to highlight the encoded information. From this encoded state, I need to decode it back to the original form. Let’s denote the decoded output as $\hat{x}$. Although the diagram may not show exact lengths between x and $\hat{x}$, it’s important to note that the output typically has the same number of nodes as the input.

Feedforward networks, such as multilayer perceptrons, are commonly used for this task, though other network configurations can also be employed to achieve autoencoding.

Now, let’s talk about representation. Suppose φ_E and φ_D are non-linear mappings where E stands for encode and D stands for decode. The mapping φ_E transforms a point in space $\mathcal{X}$ to another space $\mathcal{F}$. Specifically, if x belongs to R^D (our input space), and z belongs to R^k (the encoded space), typically k < D to ensure a compact representation that reduces noise.

The decoding mapping φ_D then maps from the space $\mathcal{F}$ back to $\mathcal{X}$. The goal is to select mappings φ_E and φ_D that minimize the reconstruction error. Start with the data vector x, apply the encoding mapping φ_E, then decode using φ_D. The output $\hat{x}$ is obtained, and the objective is to minimize the squared norm of the reconstruction error over all possible mappings φ_E and φ_D. The optimal mappings are indicated by φ_E^* and φ_D^*.

(Refer Slide Time: 18:27)



Re construction

$$\underline{\hat{x}} = \hat{\sigma}\left(\hat{\underline{W}}^T \underline{z} + \hat{b}\right)$$

Loss function $\mathcal{L}(\underline{x}, \underline{\hat{x}}) = \|\underline{x} - \underline{\hat{x}}\|^2$

$$= \Big\| \underline{x} - \underbrace{\hat{\sigma}\left(\hat{\underline{W}}^T\left(\sigma\left(\underline{W}^T \underline{z} + b\right)\right) + \hat{b}\right)}_{\hat{x}} \Big\|^2$$

One can also minimize the " average loss" puted by taking the E(·) over $\mathcal{L}(x, \underline{\hat{x}})$

For a more comprehensive understanding, one might also consider taking the expectation over all possible realizations of x within $\mathcal{X}$. While we can work with a hierarchy of layers, let’s simplify the discussion by focusing on a single hidden layer within the context of multilayer perceptrons.

The encoded vector z, which represents the encoded information, can be expressed as $\sigma(Wx + b)$. Here, σ denotes a non-linear activation function, which could be a sigmoid function or a rectified linear unit (ReLU), depending on your choice. The vector z represents the coded or latent representation of the data.

Reconstruction is relatively straightforward once we understand encoding. To decode the encoded vector, we use the inverse mapping. Specifically, the reconstructed vector $\hat{x}$ is given by $\hat{\sigma}\left(W^{\hat{T}}z + \hat{b}\right)$. Note that $\hat{\sigma}$ is not necessarily the same activation function used in the encoding phase; it can be a different activation function. Here, $\hat{W}$ and $\hat{b}$ are the learned weights and biases for the decoding process.

The loss function is designed to measure the discrepancy between the original vector x and the reconstructed vector $\hat{x}$. This is typically expressed as the Euclidean norm squared of the difference between x and $\hat{x}$. To formulate this compactly, you first perform encoding with $\sigma(Wx + b)$. Next, apply the decoding function, which involves applying $\hat{W}$, adding $\hat{b}$, and then passing through $\hat{\sigma}$ to obtain $\hat{x}$.

The loss function is given by the squared norm $|x - \hat{x}|^2$. To minimize this loss, you must account for all possible realizations of the data vector x. To achieve this, you compute the average loss by taking the expectation over all possible data vectors.

In essence, autoencoders work by learning a latent representation of the data vector through an encoding mapping to a lower-dimensional space. This is followed by decoding the encoded vector back to the original space using a decoding mapping. The goal is to minimize the reconstruction loss between the input and the reconstructed output. By optimizing the encoding and decoding mappings to minimize this loss, you effectively train the autoencoder. When considering the average loss over all data vectors, you incorporate the expectation operation.

With this foundation, we will now explore denoising autoencoders.