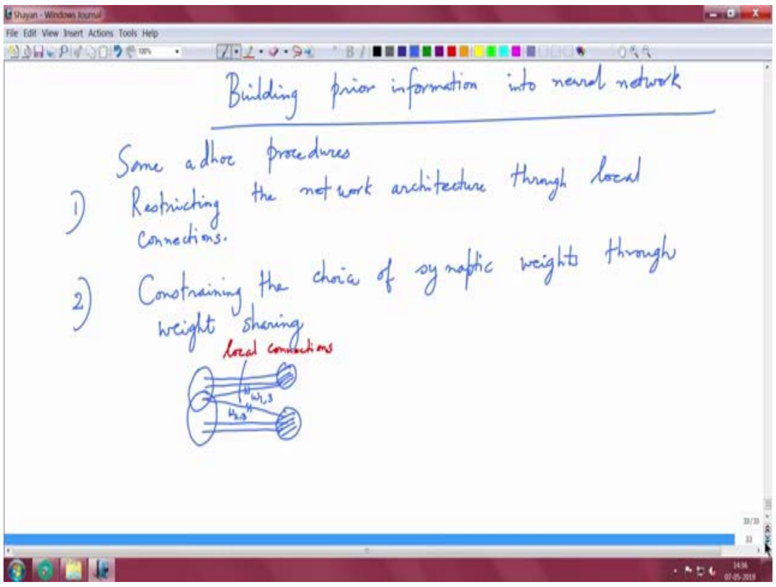
Neural Networks for Signal Processing – I

Prof. Shayan Srinivasa Garani
Department of Electronics Systems Engineering
Indian Institute of Science, Bengaluru

Lecture – 06
Prior Information and Invariances

(Refer Slide Time: 00:44)



Building prior information into neural network

Some adhoc procedures

1) Restricting the network architecture through local connections.

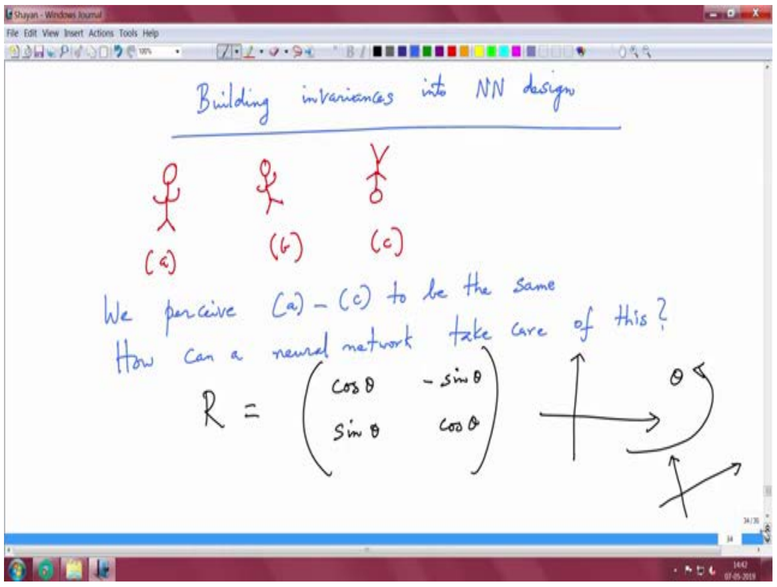
2) Constraining the choice of synaptic weights through weight sharing

Now, with these empirical or heuristic rules in mind, let's explore how we can integrate prior information into a neural network. There are several ad hoc methods to achieve this. One approach is to constrain the network architecture using local connections. By implementing specific local connections, we can restrict the network's structure.

Another method involves constraining synaptic weights through weight sharing. When discussing convolutional neural networks (CNNs), we delve into the concept of weight sharing. This technique involves assigning the same weights to different parts of the network. For instance, weights connecting one part of the network to another are shared, creating local connections.

In this setup, the shared weights facilitate the incorporation of local features across the network, thereby embedding prior information into the network's design.

(Refer Slide Time: 03:48)



Building invariances into NN design

(a) (b) (c)

We perceive (a) – (c) to be the same

How can a neural network take care of this?

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

This is largely an ad hoc decision, lacking a clear scientific foundation for incorporating prior information into neural networks. These procedures involve translating these concepts into mathematical formulations and allowing the network to learn from them.

Now, let's consider another concept: embedding invariances into the design of a neural network. How can we integrate invariances into the architecture of a neural network? Let's begin with a straightforward example.

Imagine we have the outline of a human figure, such as an Egyptian symbol. We have this shape in its standard orientation, tilted at a negative 30-degree angle, and even upside down. Despite these variations, our visual system recognizes all these images as depicting the same concept. Our visual system inherently perceives these variations as invariant representations of the same underlying object. How can we enable a neural network to similarly learn and recognize these invariances?

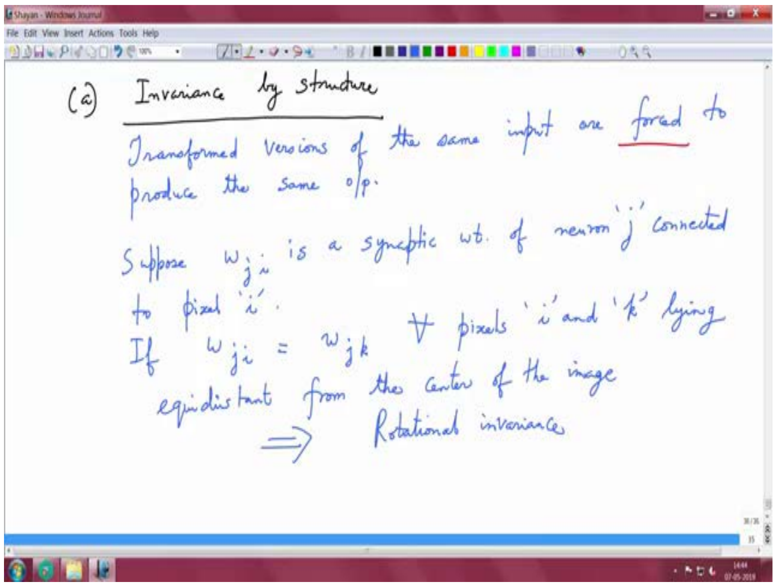
We perceive objects A through C as equivalent. How can a neural network address this? Our goal throughout this course is to enable machines to emulate human capabilities. We want machines to learn what the human brain can do. To achieve this, we need to extract and teach machines this knowledge through learning rules driven by objective functions.

Starting from basic coordinate geometry, we understand that features from A through C are interconnected by a rotational transformation R, represented as:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

This transformation rotates the coordinate axes, providing a different representation of the same features. Therefore, when extracting features, we must ensure they are rotationally invariant. This principle underscores the importance of designing the neural network's architecture in a way that naturally incorporates such invariances.

(Refer Slide Time: 07:39)



(a) Invariance by structure

Transformed versions of the same input are <u>forced</u> to produce the same o/p.

Suppose w_{ji} is a synaptic wt. of neuron 'j' connected to pixel 'i'.

If $w_{ji} = w_{jk}$ $\forall$ pixels 'i' and 'k' lying equidistant from the center of the image

$\Rightarrow$ Rotational invariance

Let's delve into two concepts of invariance: invariance by structure and invariance by training. These ideas elucidate how transformed versions of the same input are compelled to yield identical outputs.

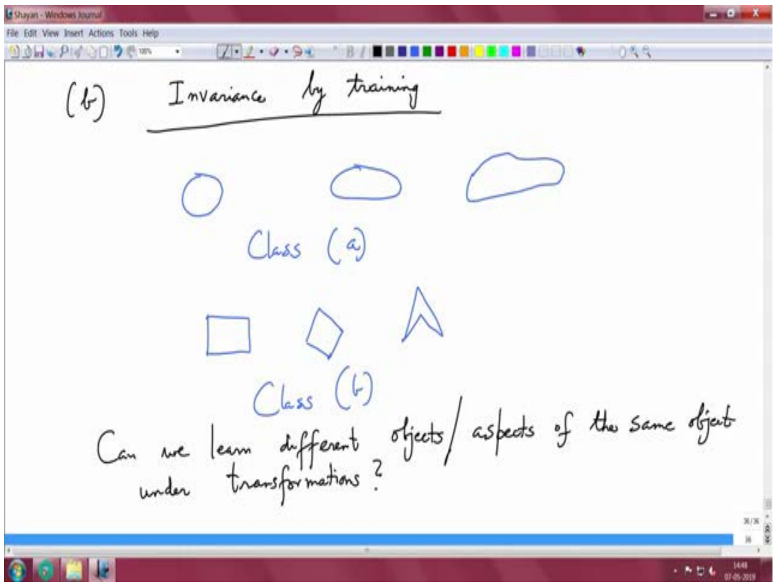
For instance, if we take object A and extract certain features from it, presenting it to the network with a desired response of 1. Similarly, if we rotate this object by an angle theta, extract its features again, and ensure it elicits the same desired response, we enforce invariance by structure. By ensuring that transformed versions of the same input produce consistent outputs, we establish this structural invariance.

Consider w_{ji}, where w_{ji} denotes the synaptic weight connecting neuron j to pixel i. This connection underscores the relationship between neurons and pixels, illustrating the implication within the network's architecture.

If $w_{ji} = w_{jk}$ for all pixels i and k equidistant from the center of the image, we naturally enforce rotational invariance within the network's design. This means that irrespective of the pixel's position relative to the center, the synaptic weights w_{ji} and w_{jk} are equal, ensuring the network treats rotated versions of the same input identically.

Additionally, we can consider other transformations such as translations and scaling. For example, when dealing with an enlarged image, we can apply local receptive field capture through masking or filtering operations. By sliding this operation across the entire image, we extract information that incorporates these invariances. Similarly, we must account for temporal or spatial translations, ensuring these transformations are seamlessly integrated into the network's framework.

(Refer Slide Time: 11:31)



(b) Invariance by training

Class (a)

Class (b)

Can we learn different objects/aspects of the same object under transformations?

Translations, rotations, and scaling are crucial transformations of the original feature set, and integrating these invariances into the neural network's design is essential. Now, let's explore another type of invariance: invariance by training.

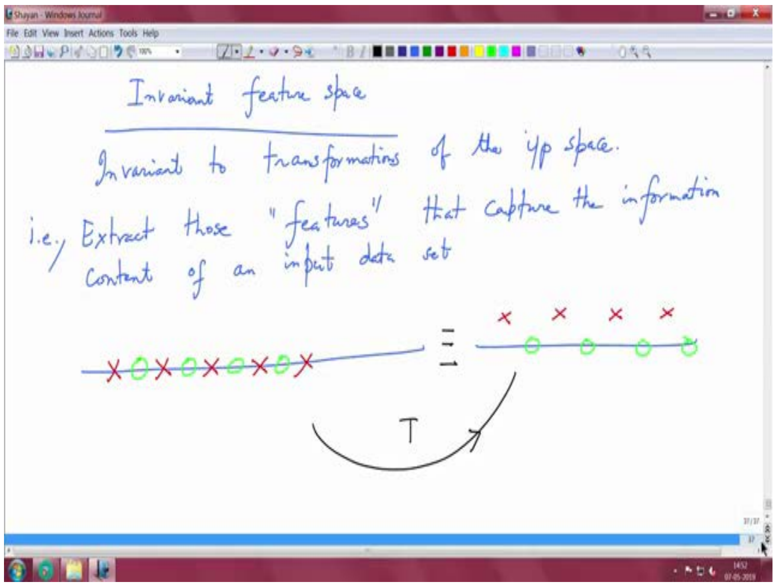
Consider two classes, A and B. Visually, we observe that class A resembles a circle, while class B appears as an ellipse or another closed curve with non-straight edges. This visual information provides prior knowledge about the nature of these classes.

If we examine class B, we observe shapes like squares and arrowheads, which are essentially quadrilaterals with straight edges. A square, for instance, can be seen as a rhombus tilted at a certain angle, while an arrowhead is another example of a quadrilateral with four sides. In contrast, class A exhibits shapes characterized by smooth curves connecting their points.

The question arises: can we teach a machine to recognize these different aspects of the same object under various transformations? This is crucial because while our human brain effortlessly identifies objects across classes A and B as equivalent, achieving this capability in machines requires deliberate design considerations.

It's imperative that we incorporate this ability into our machine learning models through training, ensuring they can perceive objects from class A and B equivalently, despite variations in their appearance due to transformations. This represents a fundamental invariance that must be learned through rigorous training processes.

(Refer Slide Time: 15:01)


Invariant feature space

Invariant to transformations of the i/p space.

i.e., Extract those "features" that capture the information content of an input data set

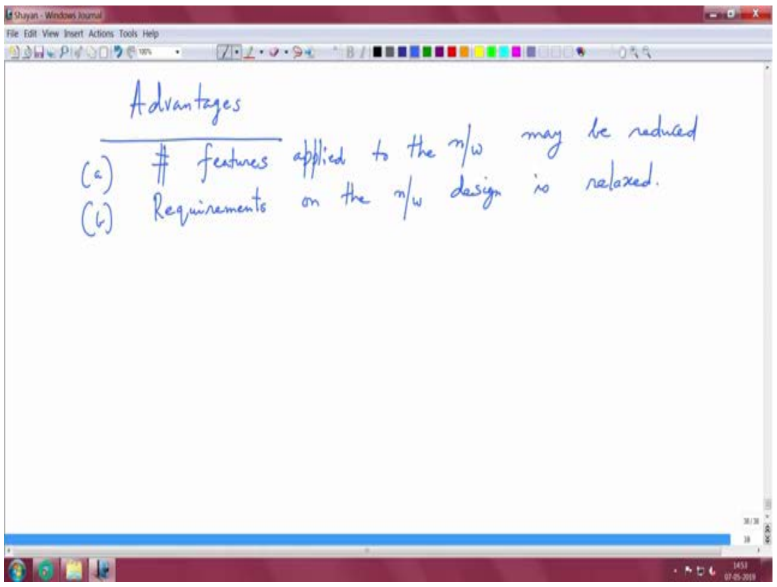
Next, we'll delve into the concept of an invariant feature space, illustrated through practical examples. An invariant feature space remains unaffected by transformations in the input space. Earlier, we explored invariances such as rotational transformations, where rotating an object doesn't alter its essence. However, these transformations aren't limited to simple linear shifts; non-linear transformations of the input space also play a crucial role in establishing invariances.

To illustrate, let's revisit our initial example of classifying odd and even integers on a line. This serves as a foundational problem where our goal is to identify features that effectively encapsulate the informational essence of a dataset.

Let's revisit the example we previously considered. Imagine we have red crosses and green circles interspersed on a graph. Our goal is to separate these groups effectively. Intuitively, if we were to lift all the red crosses vertically while leaving the green circles in place, we could achieve this separation. Essentially, this transformation would move the data points from a line to a plane, preserving their essential information without loss.

By extending this concept, we explore transformations that can maintain the integrity of data representation across different scenarios. This idea forms the basis of invariant feature spaces, where transformations enable machines to learn and adapt without compromising the underlying information content.

(Refer Slide Time: 19:20)



Advantages

(a) # features applied to the n/w may be reduced

(b) Requirements on the n/w design is relaxed.

So, how can we achieve this? This question guides us as we delve deeper into the course. What are the advantages of having invariant feature spaces? Firstly, reducing the number of features inputted into the network becomes feasible. Instead of needing a plethora of features for recognition, a subset that captures the essence under transformation suffices.

Secondly, the demands on network design are alleviated. These advantages ensure that all objects maintain their invariance under transformations, seamlessly integrated into the neural network's design. Therefore, this approach holds significant importance for solving our pattern recognition challenges.