

**Neural Networks for Signal Processing-I**  
**Prof. Shayan Srinivasa Garani**  
**Department of Electronic System Engineering**  
**Indian Institute of Science, Bengaluru**

**Lecture – 50**

**XOR Problem Revisited using RBF**

Having discussed the conditions for regularization, let's explore how we can further optimize our approach beyond the optimal solution to simplify the model, especially when dealing with a large data set. Instead of assigning a hidden unit to every point in the data set, which increases the complexity of the model, we aim to find an approximate solution. By reducing the number of hidden units, we make the model more efficient while still maintaining a high level of accuracy. This leads us to the problem of solving the Radial Basis Function (RBF) network and determining the optimal weights.

(Refer Slide Time: 03:49)

The image shows a video lecture slide with a whiteboard background. The title is "XOR Problem Revisited". The text on the slide reads: "We will consider RBF n/w as a special case of the Green's n/w." followed by "Consider the pair of Gaussian functions" and the equation  $G_i(\|\underline{x} - \underline{t}_i\|) = \exp(-\|\underline{x} - \underline{t}_i\|^2)$  for  $i = 1, 2$ . Below this, it says "Let us choose centers @  $\underline{t}_1$  and  $\underline{t}_2$ " and shows  $\underline{t}_1 = [1 \ 1]^T$  and  $\underline{t}_2 = [0 \ 0]^T$ . A red handwritten note says "At this moment, forget optimization over  $\underline{t}_1, \underline{t}_2$ ". The video player interface at the bottom shows a progress bar at 3:49 / 13:47 and the YouTube logo.

Let's delve into this process by considering the Radial Basis Function network as a specific instance of Green's function networks. We begin by examining a pair of Gaussian functions, defined as follows:

$$g(|x - t_i|) = \exp(-|x - T_i|^2) \quad \text{for } i = 1, 2$$

Here, we choose the centers  $T_1 = (1,1)^T$  and  $T_2 = (0,0)^T$ . For now, we will not focus on optimizing the choices of  $T_1$  and  $T_2$ , nor will we incorporate regularization at this stage. Instead, we assume these centers are fixed and proceed to solve the XOR problem under these assumptions.

(Refer Slide Time: 07:47)

The slide content includes the following elements:

- Equation:  $y(x) = \sum_{i=1}^2 w_i G(\|x - t_i\|) + b$
- Equation:  $y(x_j) = d_j$  where  $j = 1, 2, 3, 4; x_j \in \mathbb{R}^2$
- Table:
 

$x_j$	$d_j$
(1, 1)	0
(0, 1)	1
(0, 0)	0
(1, 0)	1
- Diagram: A neural network with two input nodes (labeled  $x_1, x_2$  and "I/p nodes"), two hidden nodes (labeled  $y_1, y_2$  and "Gaussian hidden nodes"), and one output node (labeled  $y(x)$  and "o/p node"). Weights  $w$  connect the hidden nodes to the output node, and a bias  $b$  is added to the output.

In this context, the output  $y(x)$  is given by:

$$y(x) = \sum_{i=1}^2 w_i g(|x - T_i|) + b$$

where  $w_i$  are the weights connecting the hidden units (Green's units) to the output, and  $b$  is the bias term. For the XOR problem,  $y(x_j) = d_j$ , where  $d_j$  is the desired response, and we have four data points. Let's enumerate these details:

- Data vector  $x_j$ : (1,1), (0,1), (0,0), (1,0)
- Desired response  $d_j$ : 0, 1, 0, 1

The structure of the network is as follows: We have two input coordinates  $x_1$  and  $x_2$ , as  $x_j$  is a two-dimensional vector belonging to  $R^2$ . This implies that we use two hidden units,  $\phi_1$  and  $\phi_2$ , corresponding to our Gaussian functions. The inputs connect to these hidden units, which are then combined through weights to produce the output. Given the symmetric nature of the XOR problem, we can assume weight sharing, meaning the same weight  $w$  is applied, and a bias term  $b$  is added to the output.

(Refer Slide Time: 09:33)

The video frame shows the following handwritten content:

$$G = \begin{bmatrix} 1 & 0.1353 & | & 1 \\ 0.367 & 0.3678 & | & 1 \\ 0.1353 & 1 & | & 1 \\ 0.3678 & 0.3678 & | & 1 \end{bmatrix} \quad 4 \times 3$$

$$\underline{d} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T \quad 4 \times 1$$

$$\underline{w} = \begin{bmatrix} w & w & b \end{bmatrix}^T \quad 3 \times 1$$

Annotations on the whiteboard:

- A bracket under the first two columns of  $G$  is labeled "4x2".
- An arrow points from "4 data points" to the first two columns of  $G$ .
- An arrow points from "2 centers  $t_1, t_2$ " to the first two columns of  $G$ .

Now that all elements are defined, inputs  $x_j$ , desired outputs  $d_j$ , and unknown parameters  $w$  and  $b$ , we can proceed to evaluate the Gaussian units as functions of the data points  $x$ .

With  $T_i$  specified for  $i = 1$  and  $2$ , and data points indexed by  $j$  ranging from  $1$  to  $4$ , we can formulate the Gram matrix, which will allow us to solve for the unknowns and complete the optimization.

(Refer Slide Time: 11:25)

Solve:  $w = (G^T G)^{-1} G^T d$   
3x4      4x2  
 (without regularization)

Plugging Computations done earlier

$w = \begin{bmatrix} -2.5 \\ -2.5 \\ +2.84 \end{bmatrix}$  3x1

So, let's consider the matrix setup: imagine a  $4 \times 2$  submatrix representing four data points and two centers,  $T_1$  and  $T_2$ . To accommodate the bias term, I augment this matrix by adding a column of ones, transforming it into a  $4 \times 3$  matrix. In this setup, my desired output vector,  $d$ , is

$$[0 \ 1 \ 0 \ 1]^T$$

which is a  $4 \times 1$  column vector. The weight vector, including the bias, is

$$[w \ w \ b]^T, \text{ forming a } 3 \times 1 \text{ vector.}$$

The goal is to solve the equation  $w = (G^T G)^{-1} G^T d$  to find the weights, where  $G$  is our augmented matrix. Without applying regularization, if you plug in the matrix  $G$  and vector

$d$  into this equation and perform the necessary numerical computations, you'll find that the weight vector  $w$  is approximately  $[-2.5 \quad -2.5 \quad 2.84]^T$ . This result aligns with our expectations and previous computations.

(Refer Slide Time: 12:31)

Here's a quick verification:  $G^T$  is a  $3 \times 4$  matrix, and  $G^T G$  forms a  $3 \times 3$  matrix. Multiplying this by  $G^T$ , which is  $3 \times 4$ , and then by the  $4 \times 1$  vector  $d$ , you indeed get a  $3 \times 1$  vector, precisely the format of our weight vector.

With these parameters in hand, we've effectively solved the XOR problem using an RBF network. This approach allows us to strategically choose the centers  $T_1$  and  $T_2$  rather than assigning a hidden unit to every data point. By setting up the problem this way, you can calculate the weights and biases needed to configure the RBF network to solve the XOR problem.

This example demonstrates how you can apply an RBF network to address non-linearly separable problems like XOR. It also shows how you can optimize the network by carefully

selecting the centers of the Gaussian functions instead of overcomplicating the model with an excessive number of hidden units.

Hopefully, this example clarifies how you can structure the network and solve for its parameters based on your data and desired output. If you wish to introduce regularization constraints, you can easily incorporate them into the solution process. With this foundational example, you should be better equipped to handle more complex scenarios involving large data sets, where various optimization techniques can be applied. We will stop here for now and in the next session, we will delve into solving regression problems under regularization constraints.