Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani

Department of Electronic System Engineering

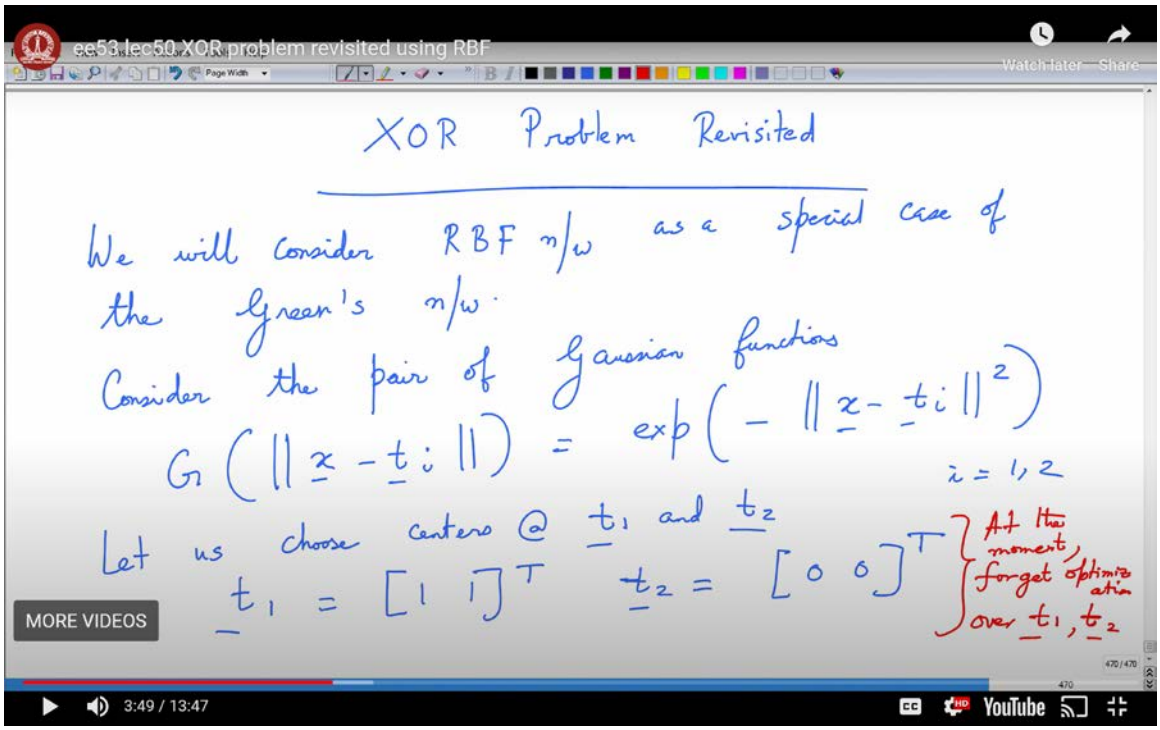
Indian Institute of Science, Bengaluru

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)

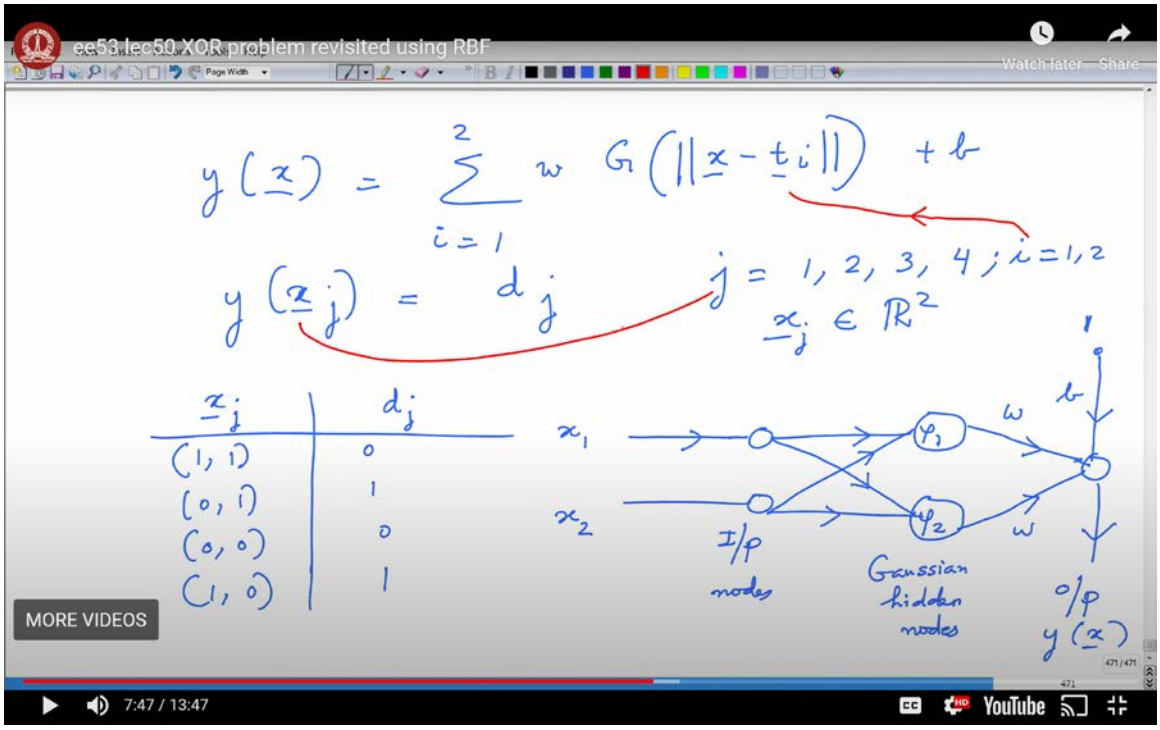


Let's delve into this process by considering the Radial Basis Function network as a specific instance of Green's function networks. We begin by examining a pair of Gaussian functions, defined as follows:

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

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)



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

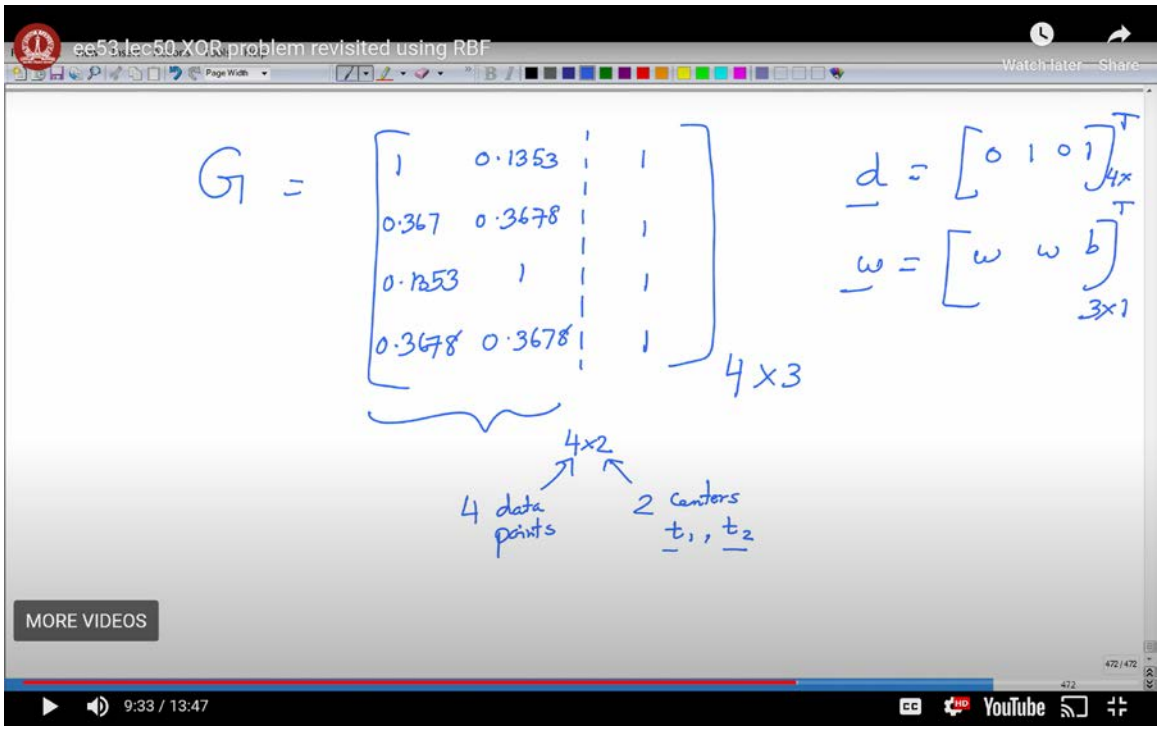
$$y(x) = \sum_{i=1}^{2} w_i g(|x - T_i|) + b$$

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, φ_1 and φ_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.

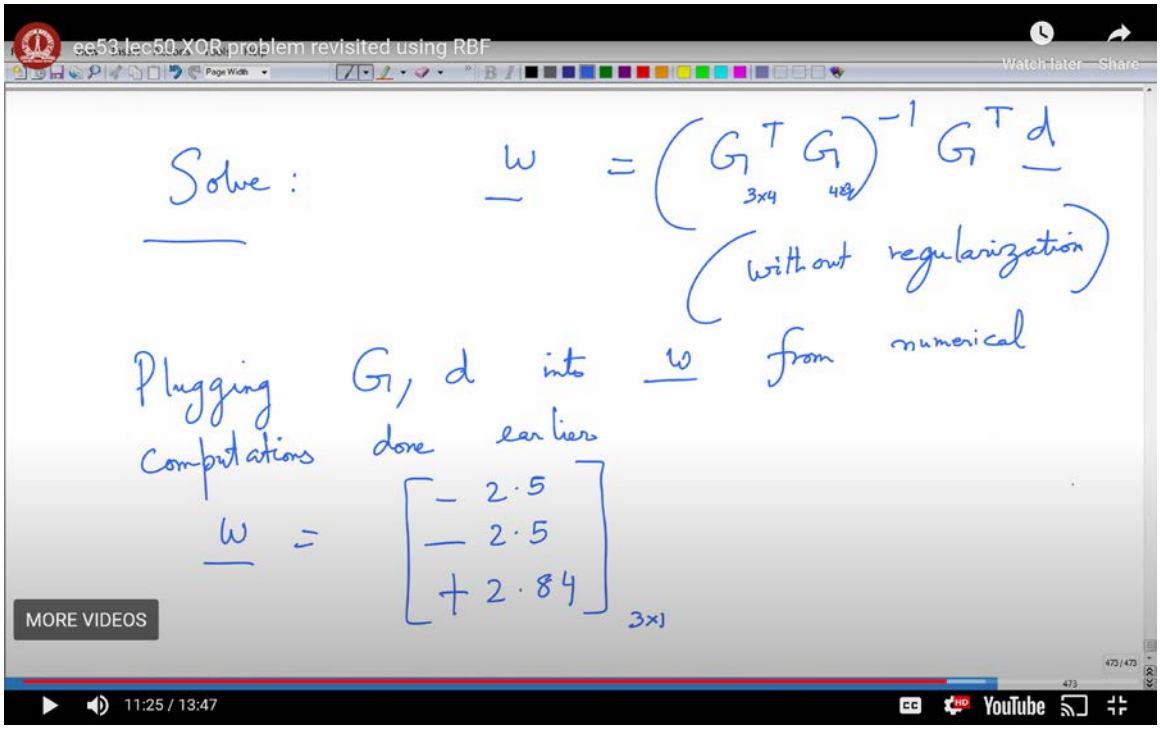
(Refer Slide Time: 09:33)



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)



So, let's consider the matrix setup: imagine a 4 × 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 × 3 matrix. In this setup, my desired output vector, $\boldsymbol{d}$, is

$$\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T$$

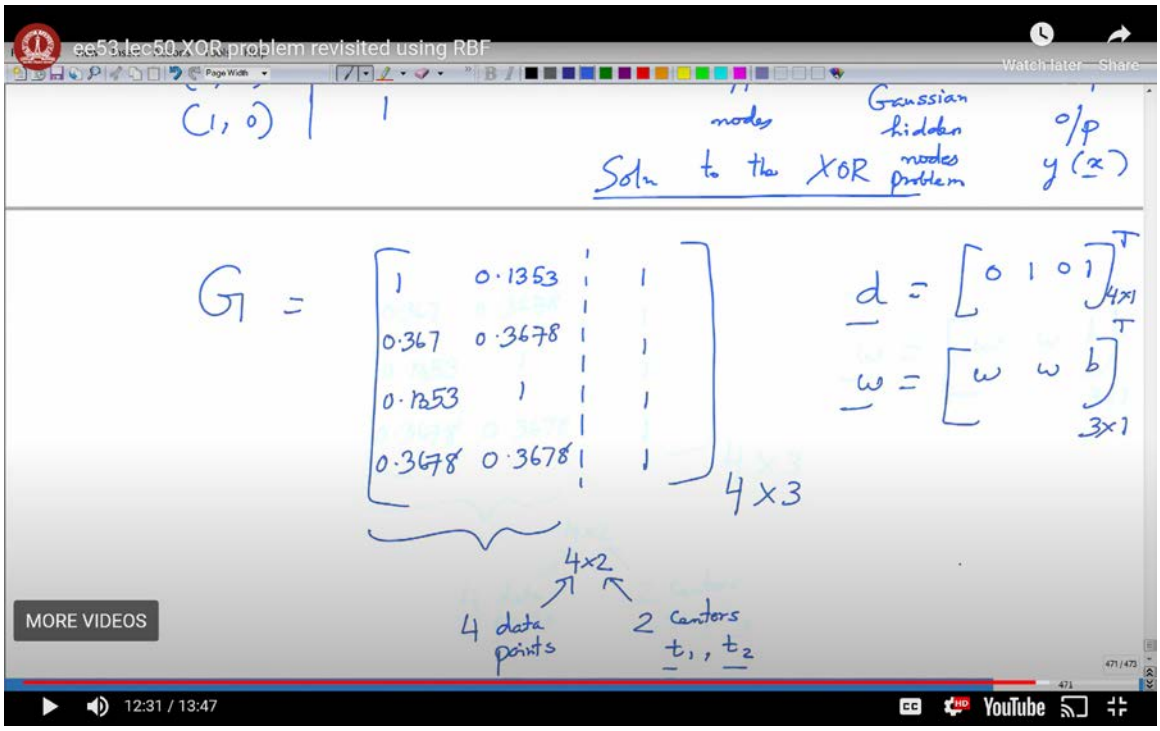
which is a 4 × 1 column vector. The weight vector, including the bias, is

$\begin{bmatrix} w & w & b \end{bmatrix}^T$, forming a 3 × 1 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 $\boldsymbol{G}$ and vector

$\boldsymbol{d}$ into this equation and perform the necessary numerical computations, you'll find that the weight vector w is approximately $\begin{bmatrix} -2.5 & -2.5 & 2.84 \end{bmatrix}^T$. This result aligns with our expectations and previous computations.

(Refer Slide Time: 12:31)



Here's a quick verification: G^T is a 3 × 4 matrix, and $G^T G$ forms a 3 × 3 matrix. Multiplying this by G^T, which is 3 × 4, and then by the 4 × 1 vector d, you indeed get a 3 × 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.