Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani

Department of Electronic System Engineering

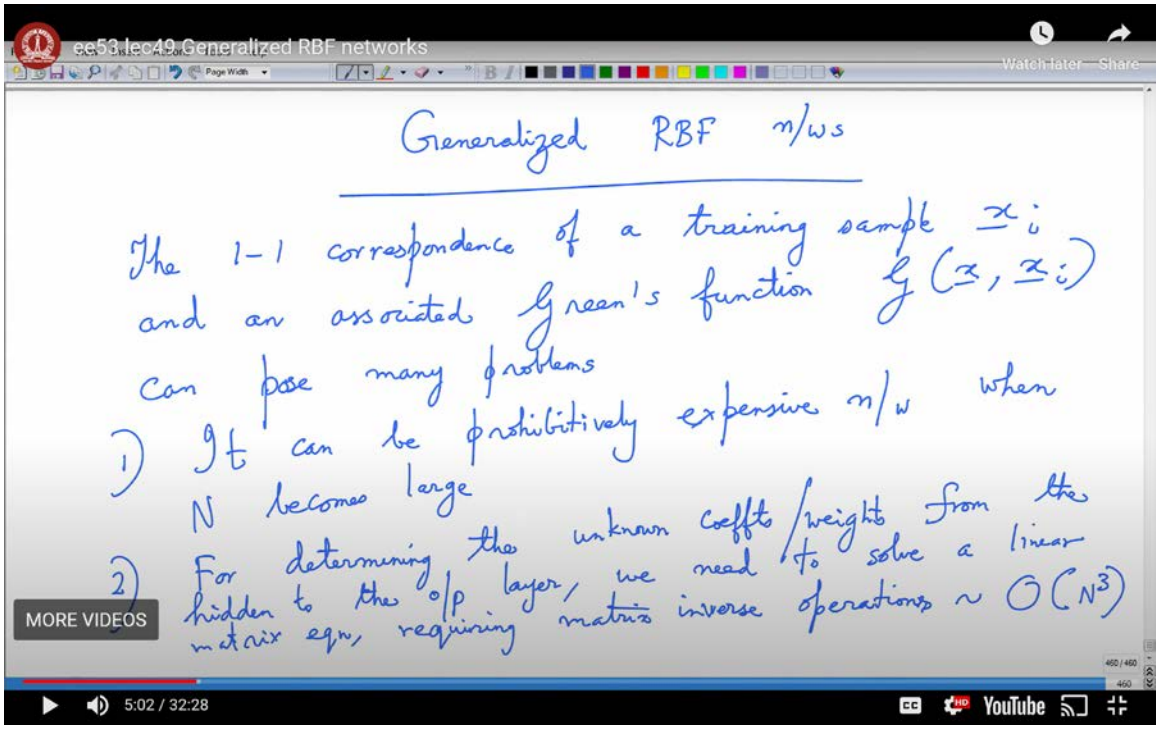
Indian Institute of Science, Bengaluru

Lecture – 49

Generalized RBF Networks

In this module, let's delve into generalized Radial Basis Function (RBF) networks as part of our course. At this point, you might be wondering, especially after seeing the architectural sketch of the Green's function-based regularization network, why we need a Green's function unit $g(x, \zeta_i)$ associated with every data sample ζ_i. In practical scenarios, datasets can be overwhelmingly large, potentially consisting of millions or even tens of millions of data points.

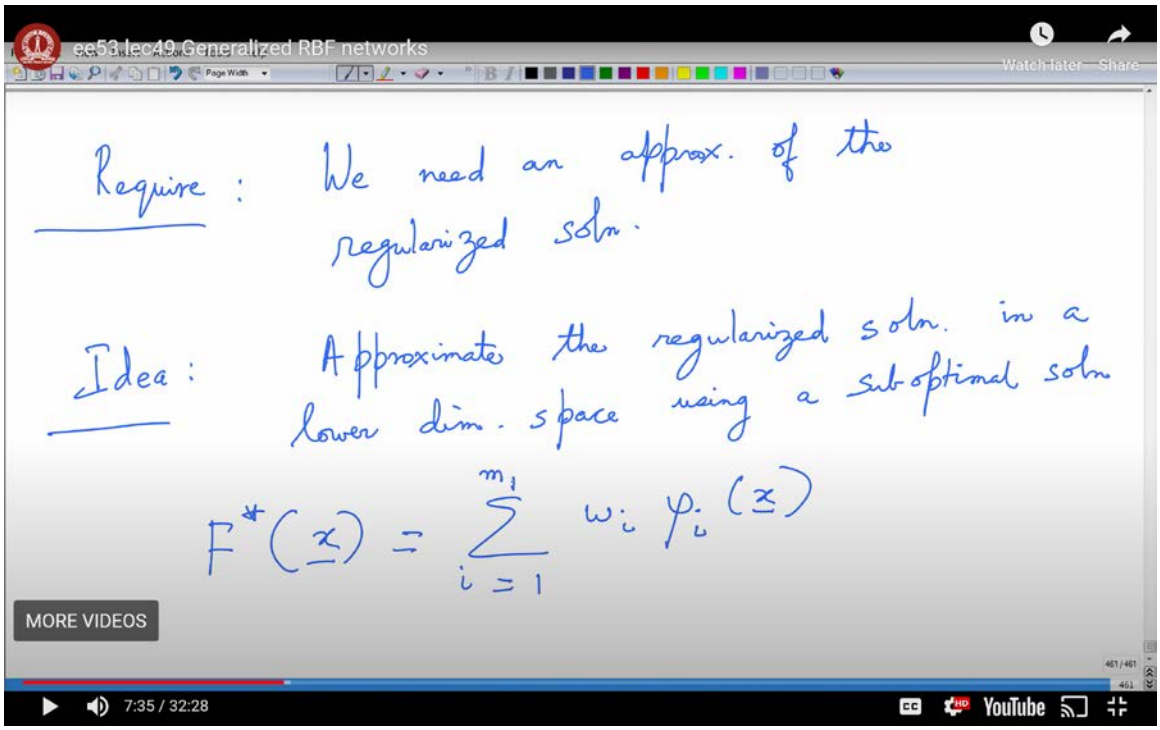
(Refer Slide Time: 05:02)



If we were to follow this approach strictly, it would necessitate having millions or tens of millions of nodes in the hidden layer, where each node acts as a Green's function unit. This is not just impractical; it’s outright prohibitive.

We need to explore ways to restrict the size of the network, which is a crucial aspect of our study. So, how does the network behave under these constraints, and how do we go about solving for the optimal weights? Let’s examine this in detail.

(Refer Slide Time: 07:35)

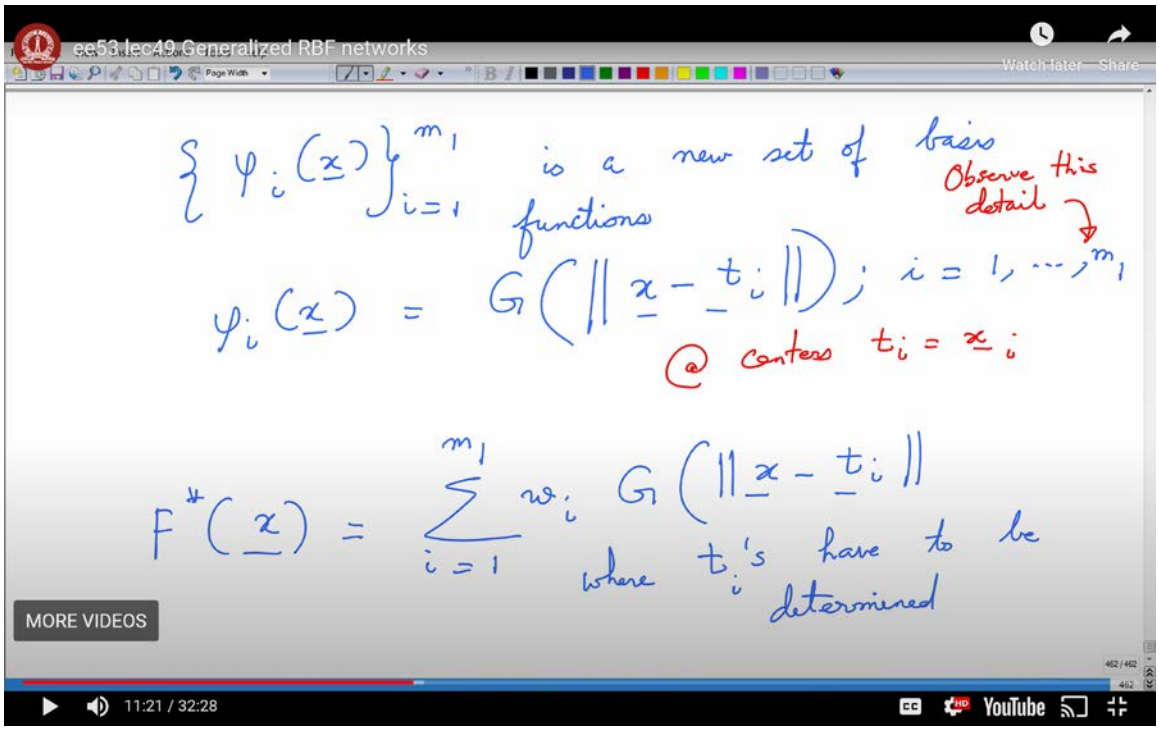


The one-to-one correspondence between each training sample ζ_i and its associated Green's function $g(x, \zeta_i)$ can introduce significant challenges, particularly on the computational front. These challenges include both memory usage and computational demands. First, it can result in an excessively expensive network, especially as n becomes large. Second, determining the unknown coefficients or weights from the hidden layer to the output layer requires solving a linear matrix equation. Solving this matrix equation involves inverting a matrix of size $n \times n$, which has a computational complexity on the order of $O(n^3)$. This complexity can become quite overwhelming and prohibitive.

Moreover, there's the issue of matrix ill-conditioning, which is another critical concern. Consider the situation where you need to solve for a weight vector x in the equation $Ax = B$, where A is your matrix and B is your desired response. Solving for x requires computing $A^{-1}B$, but inverting this matrix can be a significant challenge if the matrix is large. Even more problematic is the possibility that the matrix might be ill-conditioned, meaning its inversion is unstable or inaccurate, depending on how the matrix is formed. This ill-conditioning can make the problem even more complex and demands special handling.

Given these challenges, additional regularization becomes necessary. What we need, in essence, is an approximation of the regularized solution. This approach forms the core of the idea we are developing in this context.

(Refer Slide Time: 11:21)

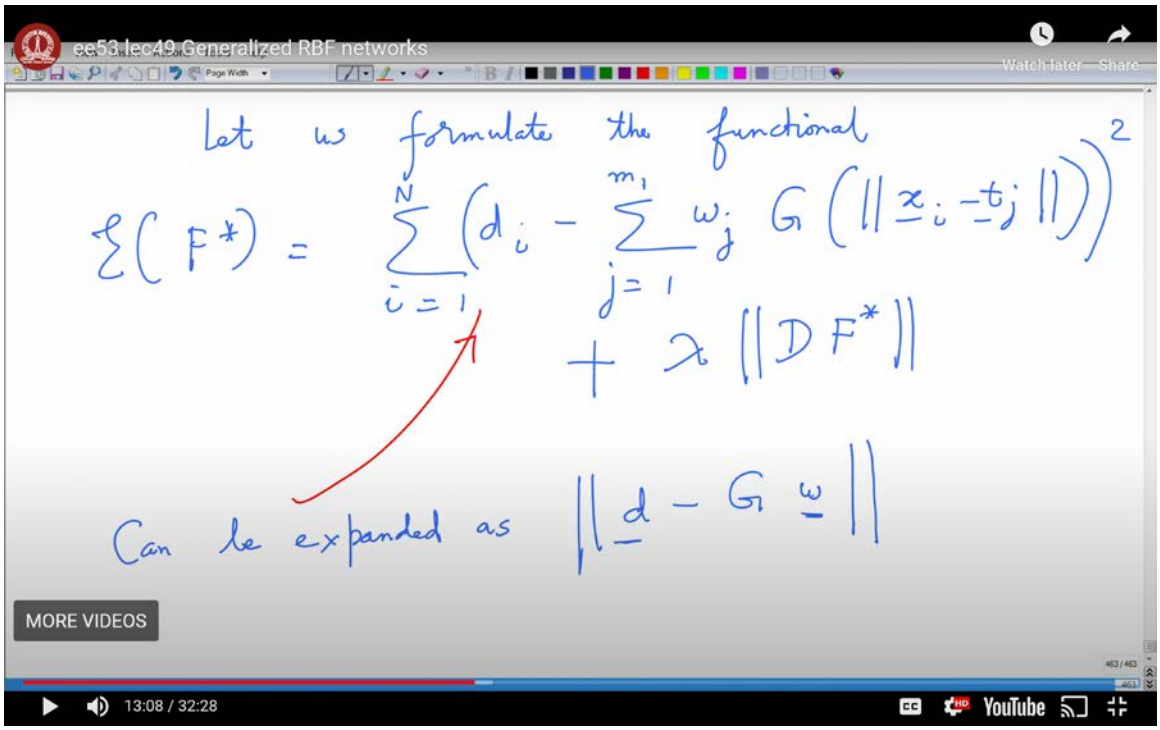


We aim to approximate the regularized solution within a lower-dimensional space, potentially using a suboptimal solution. What this entails is expressing $f^*(x)$ as a linear combination of a new set of basis functions, denoted by $\varphi_i(x)$. While it remains a linear

combination, this approach leverages a much smaller set of basis functions within an M_1-dimensional space, allowing us to approximate the function more efficiently.

If you recall, the original function f(x) was constructed using all the training samples, where each sample had an associated Green's function. However, this method is prohibitively complex. When solving this, particularly when forming and inverting the Gram matrix, issues such as ill-conditioning can arise. Despite efforts to impose positive definiteness and other desirable properties, depending on the nature of your data points, the matrix may still be ill-conditioned.

(Refer Slide Time: 13:08)



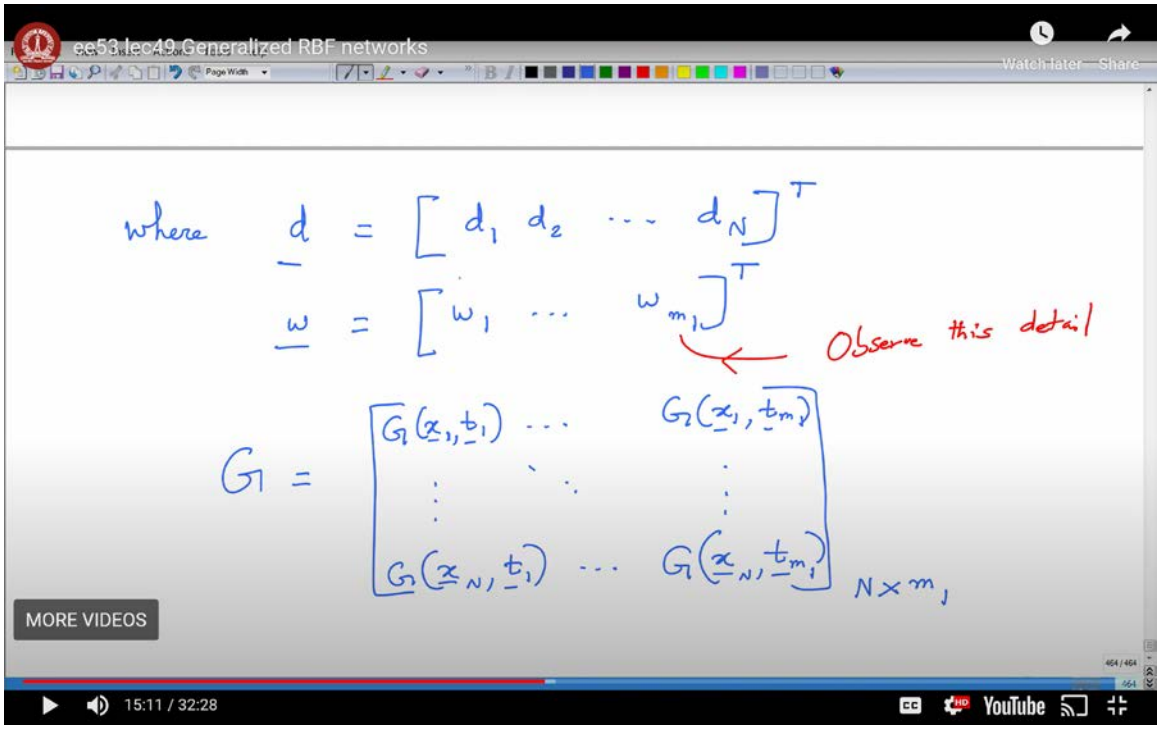
Therefore, we seek to approximate the optimal solution to the regularization problem within a lower-dimensional space, where the problem becomes more tractable. Let's move forward with this idea.

In this new formulation, the functions $\varphi_i(x)$ for i = 1 to M_1 represent a new set of basis functions. As basis functions, they must be linearly independent. We define $\varphi_i(x)$ as $g(x - t_i)$, where i ranges from 1 to M_1.

It's crucial to notice the change here: instead of having N units, we now have M_1 units. These units are centered at t_i, where t_i equals x_i for some choice of i from 1 to M_1. Now, we express $f^*(x)$ as a summation:

$$f^*(x) = \sum_{i=1}^{M_1} w_i g(|x - t_i|)$$

(Refer Slide Time: 15:11)



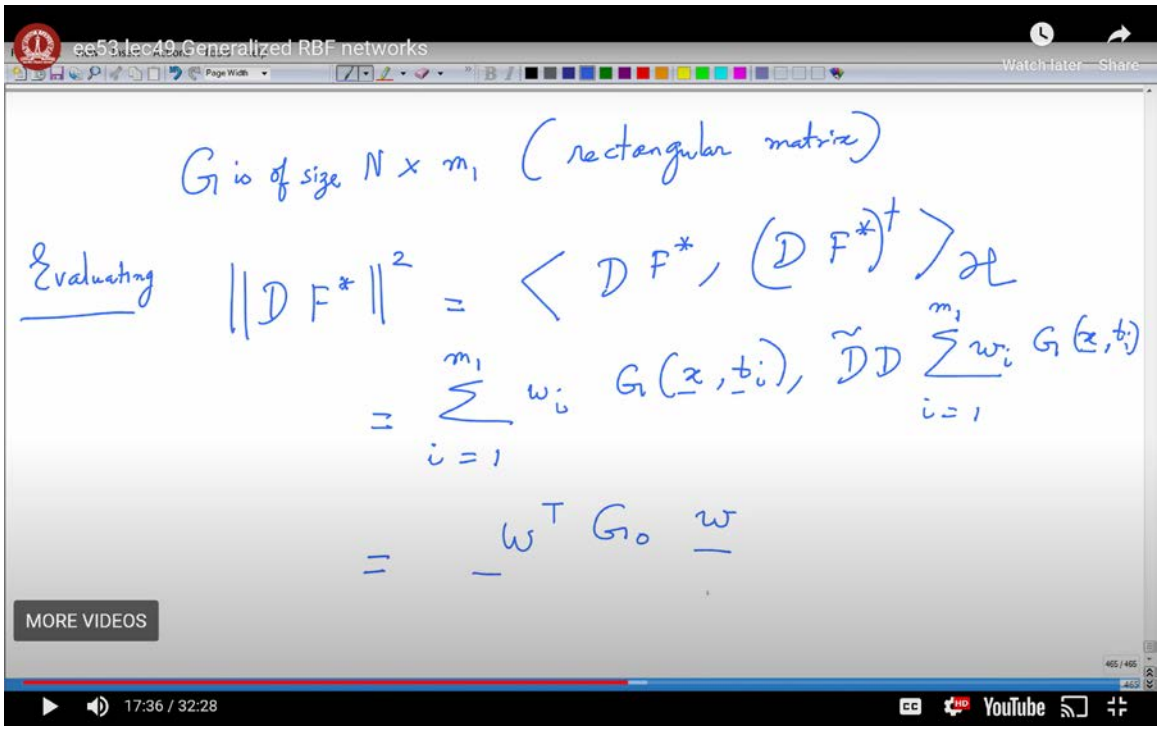
Here, the centers t_i must be determined. If we set $t_i = x_i$ for i = 1 to N, we revert to the original problem. However, to avoid this, we restrict our solution to i = 1 to M_1.

Let's formulate the functional. We define it as follows:

$$E(f^*) = \sum_{i=1}^{N} \left(d_i - \sum_{j=1}^{M_1} w_j g\left(|x_i - t_j|\right) \right)^2$$

+subject to the regulatory conditions where a differential operator acts on f^*

(Refer Slide Time: 17:36)



This first term can be expanded as the Euclidean norm of two vectors, which can be expressed as:

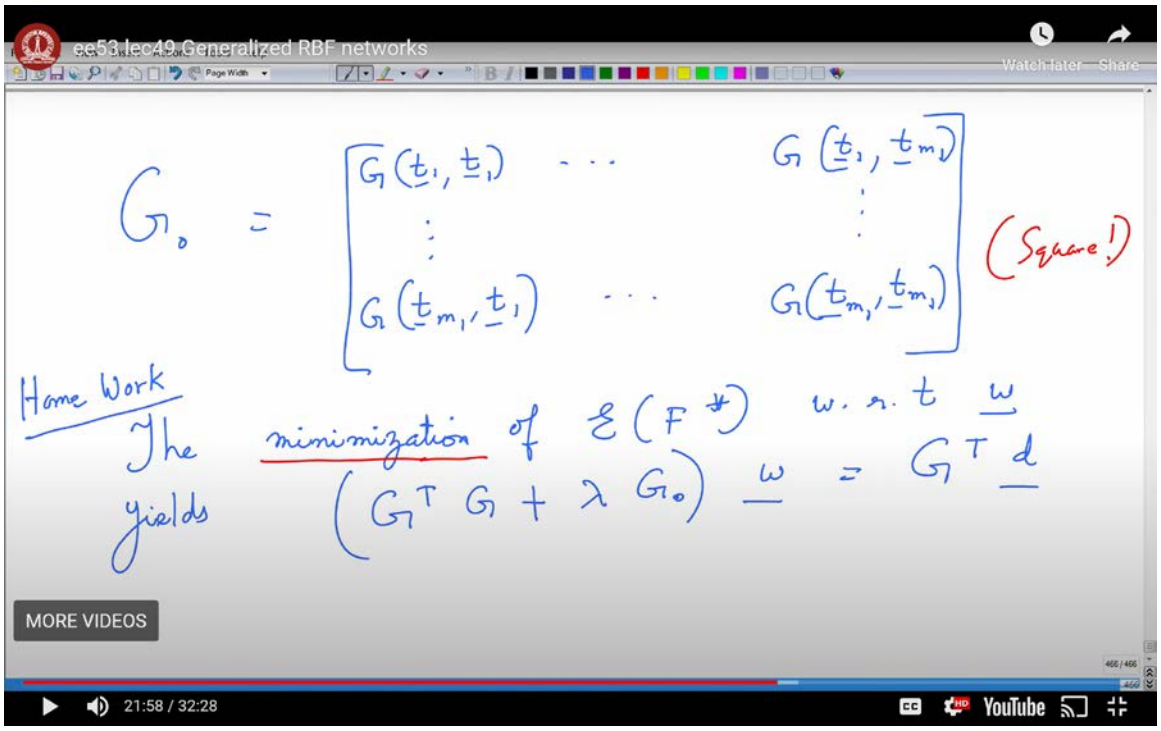
$$E(f^*) = |d - Gw|^2$$

Where d is a vector, w is a vector of weights, and G is your matrix. Let me define these variables clearly:

- d is a column vector: $d = [d_1, d_2, \ldots, d_N]^T$
- w is the vector of weights corresponding to the M_1 units.
- G is the Gram matrix with dimensions N × M_1, where each element is $G_{ij} = g(|x_i - t_j|)$.

To expand on the matrix G, it is a rectangular matrix, no longer symmetric, with N rows and M_1 columns, representing the connections between the N data points and the M_1 centers.

(Refer Slide Time: 21:58)



So, to summarize, we have redefined the problem to operate within a reduced-dimensional space, making the computation more manageable while still approximating the regularized solution effectively. This approach ensures that we bypass the complexities of the original, larger problem and achieve a more efficient computation.

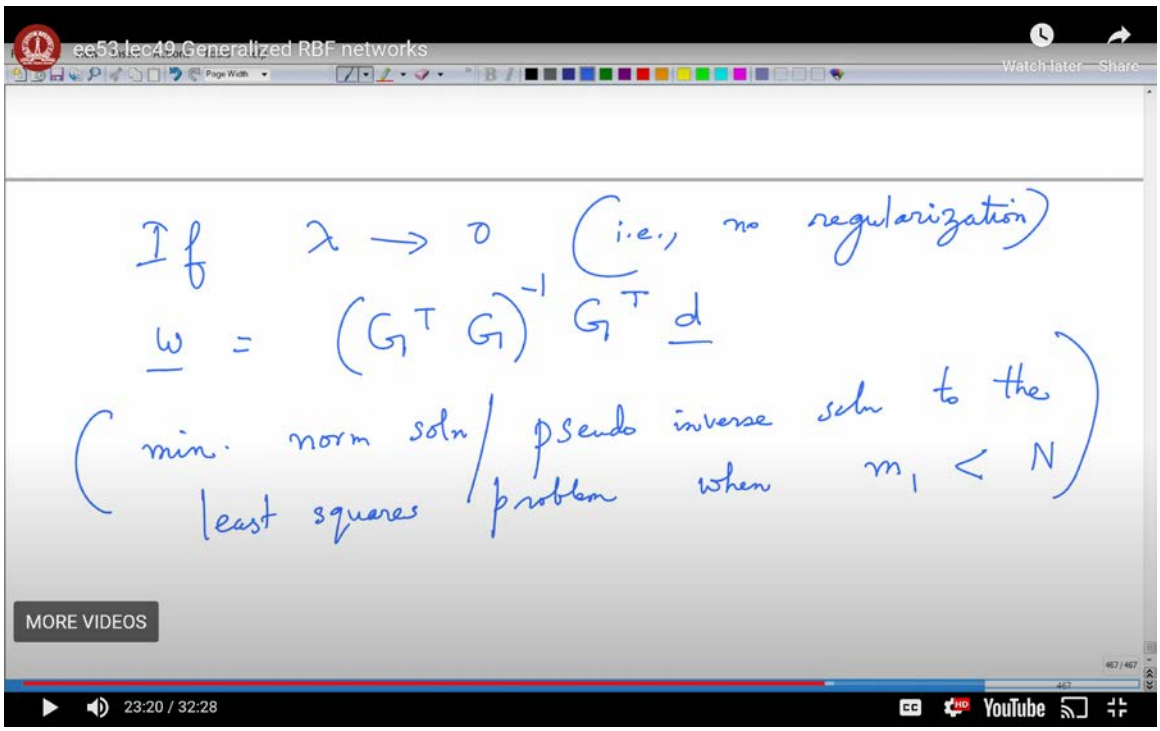
Let's delve into the evaluation of the norm of d applied to f^*, where d represents the differential operator acting on the approximating function. Essentially, this is the inner product of d with f^* and its adjoint in the Hilbert space H, where this inner product is defined. This expression can be simplified as:

$$\sum_{i=1}^{M_1} W(x_i) \cdot g(x, t_i) \cdot \left(d^\dagger d\right) \sum_{i=1}^{M_1} W(x_i) \cdot g(x, t_i)$$

This can be interpreted as a quadratic form $W^T G_0 W$, where G_0 is a square matrix, commonly known as the Gram matrix, and is composed of points evaluated at t_i and t_j.

Now, let's break this down: We have two primary terms that we've expanded in matrix form. The regularization constraint corresponds to this quadratic form $W^T G_0 W$, and the approximation error can be represented as the Euclidean norm |D - GW|. Our goal is to minimize this Tikhonov functional, which involves minimizing the Euclidean norm subject to these regularization constraints. I'll leave this as an exercise to work through.

(Refer Slide Time: 23:20)



Minimizing the Tikhonov functional $E(f^*)$ with respect to the weights W yields the solution:

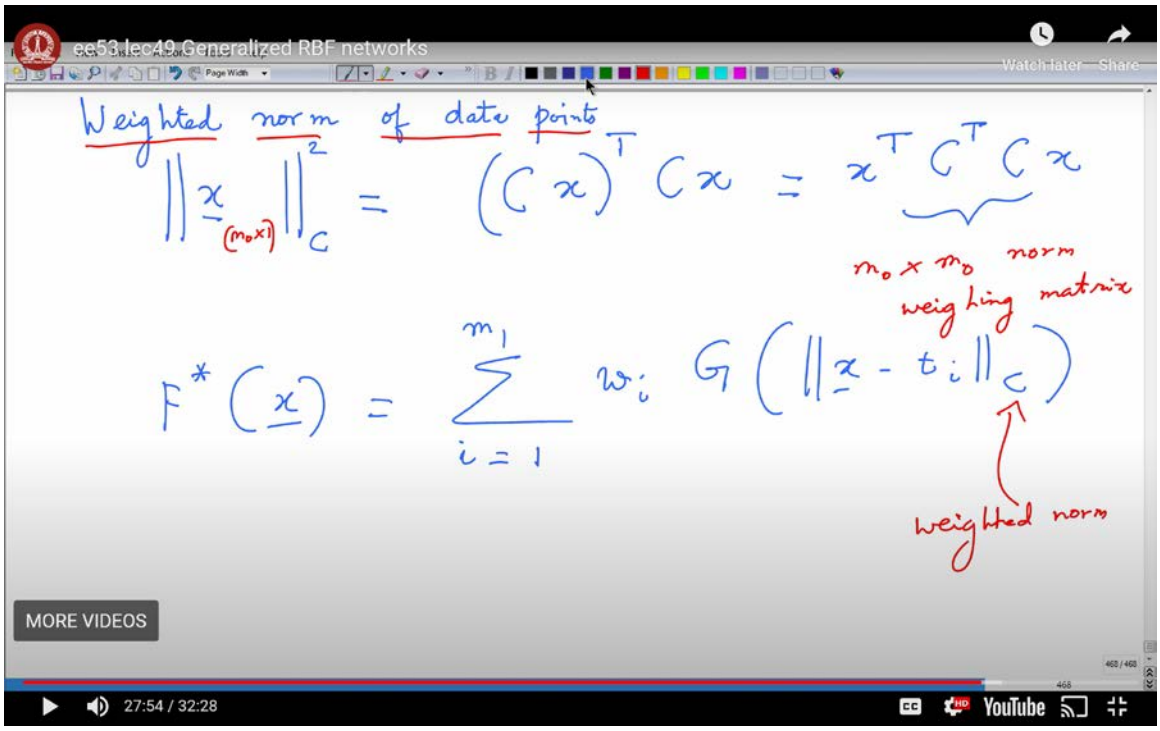
$$(G^T G + \lambda G_0)W = G^T D$$

When λ tends to zero, the solution W essentially becomes the pseudoinverse or the minimum norm solution to the overdetermined least squares problem, particularly when M_1 is less than N. This is a valuable insight, and I encourage you to explore this further as homework. You can refer to my video lectures on matrix calculus, where I cover how to differentiate scalars with respect to vectors, and matrices with respect to vectors. These

matrix calculus operations can be straightforwardly applied when expressing this functional in matrix form.

Once you carry out the matrix calculus, the solution becomes evident, as I've outlined. Given our definitions of the matrices G, G_0, and the vectors W and D, you'll be able to compute the optimal solution.

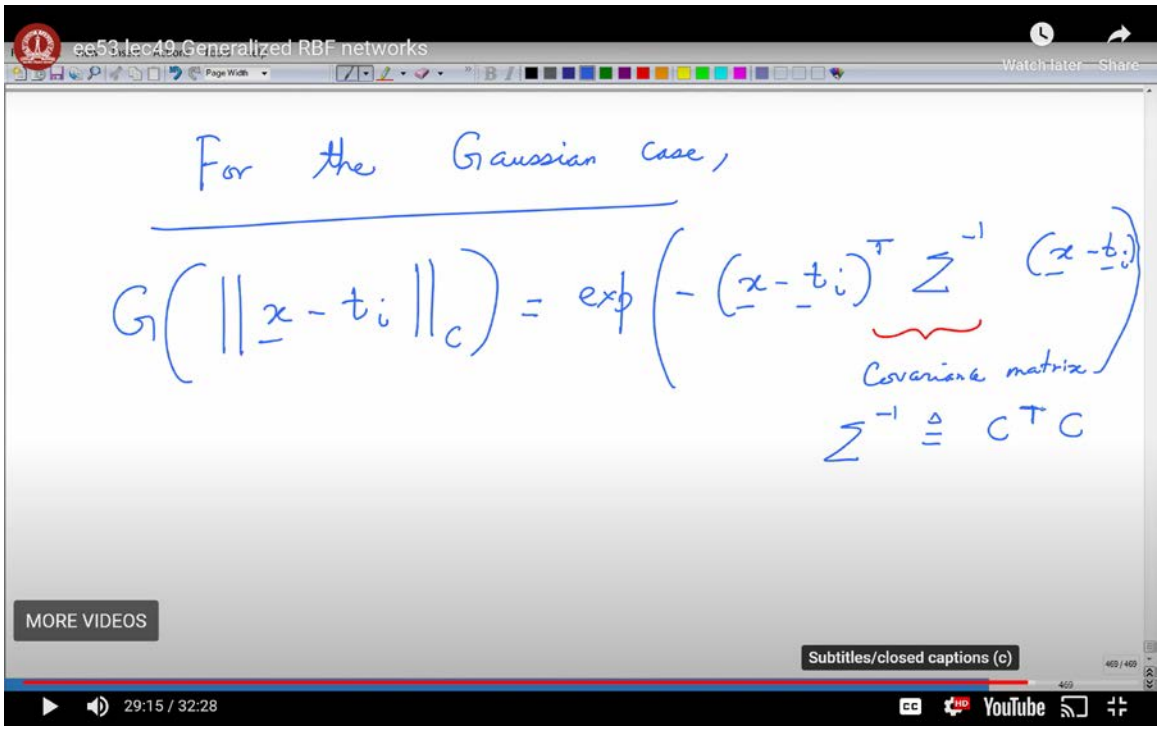
(Refer Slide Time: 27:54)



However, the critical takeaway here lies in the conclusions we can draw. If λ tends to zero, meaning no regularization is applied, then W simplifies to $(G^T G)^{-1} G^T D$, which represents the minimum norm solution or the pseudoinverse for the least squares problem. This situation typically arises when M_1 is less than N, meaning we are performing some data fitting and seeking an approximate solution. Under these conditions, this reduces to the minimum norm solution, which is what we achieve here. Depending on the value of λ, we adjust this by λ G_0, which introduces the regularization effect into the solution for the weight vector W.

Now, let's consider an important interpretation. Sometimes, data itself may be weighted. For instance, when dealing with Principal Component Analysis (PCA), we work with covariance matrices. In a multivariate distribution or density context, the data vectors may be scaled by these covariance matrices, which essentially act as weights for these data vectors in a probabilistic sense. Similarly, you might want to weight data points using a weighting function, especially when not all data points are equally relevant. Depending on the problem, certain clusters of points may deserve more weight than others in your cost function. This weighting process is highly context-dependent.

(Refer Slide Time: 29:15)



Let's now explore how we can set up a scenario where data points are weighted appropriately.

In this context, I can define a norm by introducing a weighting function, denoted by C, which effectively allows data points x to be weighted according to this function. We assume C to be a matrix, and the norm of x, weighted by C, can be expressed as:

$$\text{Norm of } x_C^2 = C \times x^T \times C \times x$$

In quadratic form, this is written as:

$$x^T \times C^T \times C \times x$$

Here, $C^T \times C$ serves as the norm-weighting matrix, which is an $M_0 \times M_0$ matrix. Assuming the data vector x is of dimensions $M_0 \times 1$, this formulation makes sense.

Now, let's explore an example. Consider the approximating function $f^*(x)$, which can now be expressed as:

$$f^*(x) = \sum_{i=1}^{M_1} w_i \cdot g(x - t_i)$$

In this expression, the norm used in the Green's function is a weighted norm. By weighting the data points, the approximating function is expressed in terms of the weighted norm, with the Green's function reflecting the weighted norm of the data points rather than just the standard norm. This is one example.

A more familiar example is based on the Gaussian case. Here, the Green's function can be expressed over the weighted norm of the data points as:

$$\exp\left(-\frac{1}{2}(x - t_i)^T \Sigma^{-1}(x - t_i)\right)$$

This is where σ represents the familiar covariance matrix, as seen in Gaussian distributions. In the context of weighting, we can think of σ^{-1} as analogous to $C^T \times C$, similar to what we discussed earlier.

With these forms in place, we can set up our Tikhonov functional. Once we have expressed the functional in terms of matrices and vectors, we can apply straightforward matrix calculus to derive the optimal weight vector. However, this is just one detail in the broader context.

First, we establish the Tikhonov functional in its original form, incorporating regulatory conditions to address issues like overfitting or the need to minimize the number of

parameters required to approximate the regularized solution. From this, we further approximate the optimal solution to obtain a suboptimal solution by following this specific approach.

This essentially concludes our discussion on Green's functions and neural networks based on regularized Green's functions, as well as the further optimizations achievable by incorporating a weighting matrix into the data.

At this point, you might wonder how to determine the centers t_i. There are two practical approaches to this. One option is to use an external clustering procedure driven by some optimality condition to solve for the best centers. Alternatively, you can set up optimization conditions directly over these t_i values, derive the corresponding equations, and solve for them based on the optimality criteria, aiming to minimize the functional cost.

These are two viable and practical methods. Whether you choose to adaptively solve for the centers or use a different algorithm to arrive at clusters (i.e., the t_i values), this decision ultimately depends on your specific approach to the algorithm. It is subjective and based on your preferences.

Whichever method you choose, the general framework we've discussed, bringing in regulatory conditions, is crucial. I hope you find this framework valuable in your research or graduate studies.