Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani

Department of Electronic System Engineering

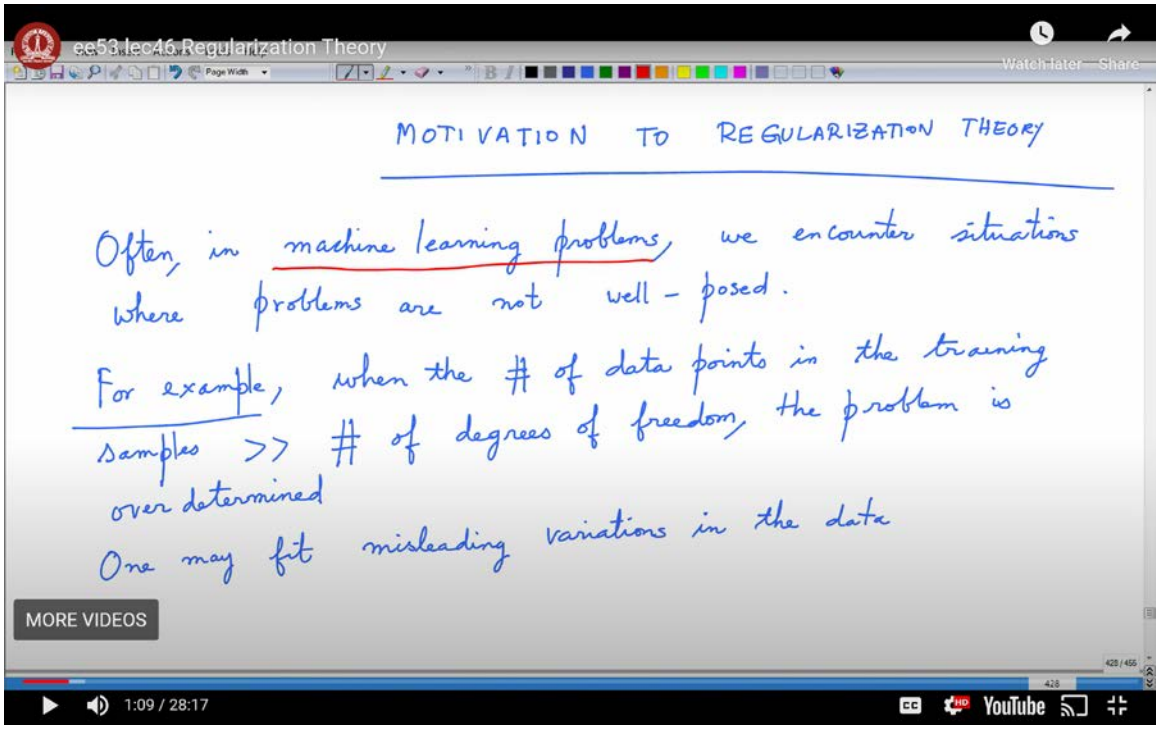
Indian Institute of Science, Bengaluru

Lecture – 46

Regularization Theory

Let's begin by diving into the motivation behind regularization theory, understanding the concepts of well-posed and ill-posed problems, and exploring why regularization is essential. We will also discuss how to modify our functional to include a regularization term and how to approach solving these problems.

(Refer Slide Time: 01:09)

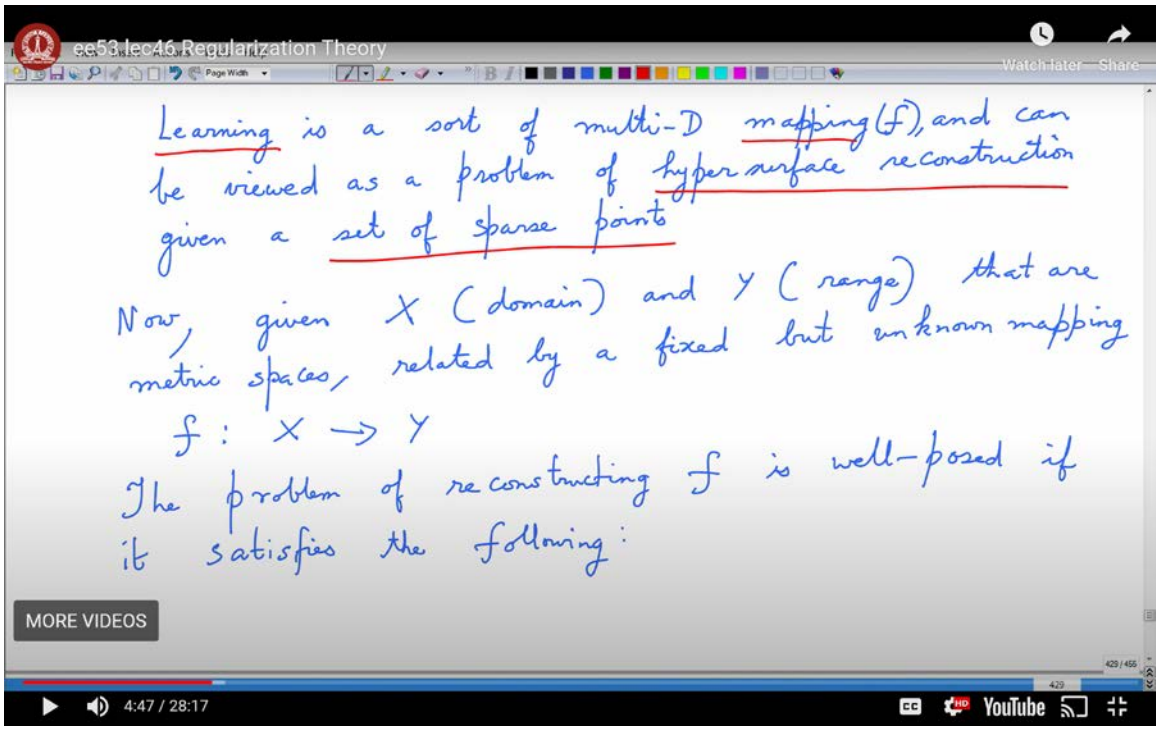


Regularization theory stems from the practical challenges we encounter in machine learning, where problems are not always well-posed. To grasp this, let's start with the concept of an overdetermined system, which occurs when the number of data points in the

training samples far exceeds the degrees of freedom. Imagine you have a system of linear equations: for instance, two simultaneous equations like x + y = 5 and x - y = 3. With two equations and two variables, you can solve for x and y precisely.

However, in more complex scenarios, you might encounter a situation where you have fewer equations but a vast amount of data samples, or fewer degrees of freedom in the parameters that you need to estimate in the functional representation of the data. In such cases, the system is overdetermined, and this isn't limited to linear forms. The issue with overdetermined systems is that they can lead to fitting misleading variations in the data, which is a clear sign of an ill-posed problem.

(Refer Slide Time: 04:47)

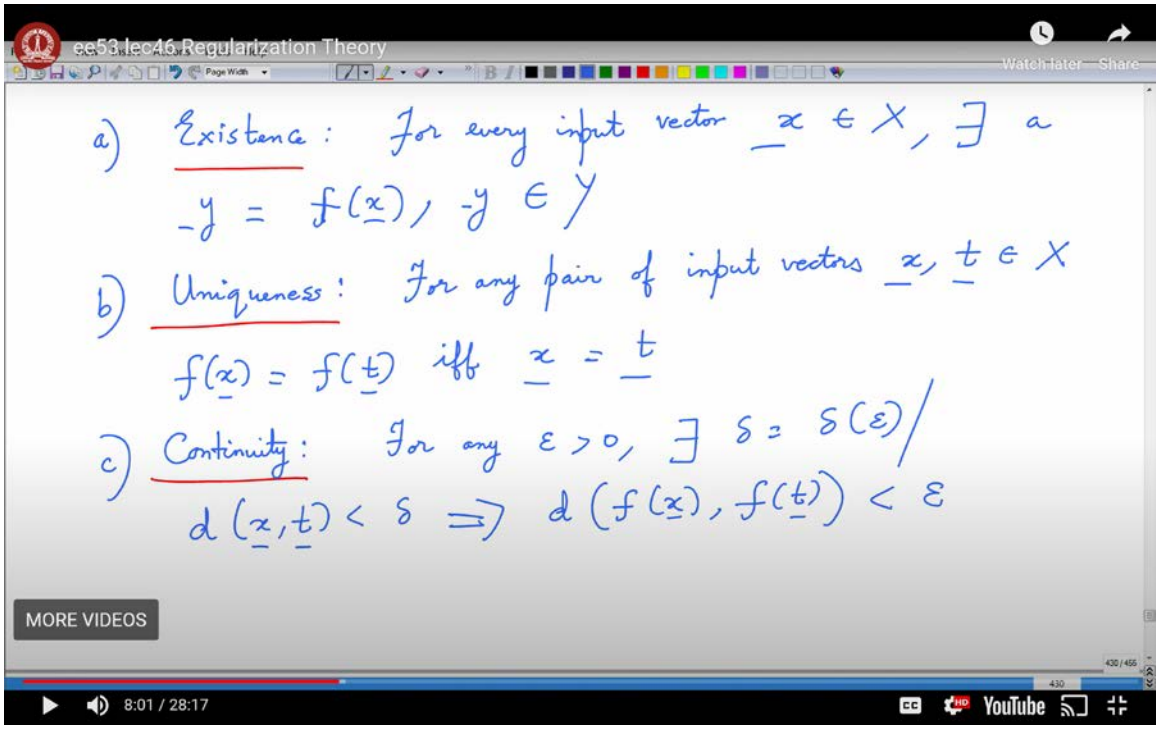


Now, let's think about learning, not in the broad, qualitative sense, but in the mathematical context of machine learning or neural networks. Here, learning can be considered a multidimensional mapping F, which can be viewed as a problem of hyper-surface reconstruction given a set of sparse data points.

To break this down: imagine you're given a domain X where your data points exist. This domain could be a multi-dimensional space, and the range, Y, could be either a vector or a scalar. For instance, if you want to associate a label with a data point, it might be a scalar, but it could also be a vector in a broader context.

Now, you have pairs of data points, x_i and y_i, where x belongs to the domain and y belongs to the range. These pairs are connected by a fixed but unknown mapping. A helpful analogy would be to think of an image of a cat. You extract feature vectors from this image and aim to associate these feature vectors with a label, perhaps "1" for a cat and "2" for a dog. This process highlights the importance of understanding the underlying mapping and the role of regularization in preventing the model from fitting to misleading variations or noise in the data.

(Refer Slide Time: 08:01)



Let's continue exploring the concept of reconstructing a multidimensional mapping, focusing on how it relates to well-posed problems and the role of regularization.

Consider this scenario: you take an image of a dog, extract its feature vectors, and then attempt to associate these vectors with a label. Essentially, the learning problem is about mapping f from x (the feature vectors) to y (the label), establishing this association. We assume that x and y exist within metric spaces.

The challenge of reconstructing this multidimensional mapping can be viewed as a hypersurface reconstruction problem over a sparse set of points. For instance, let's say the domain comprises a 20-dimensional space, R^{20}, meaning our feature vectors exist in a 20-dimensional space. Even if we have about a million data points, this is still a sparse sampling within such a vast space, making the reconstruction problem quite intricate.

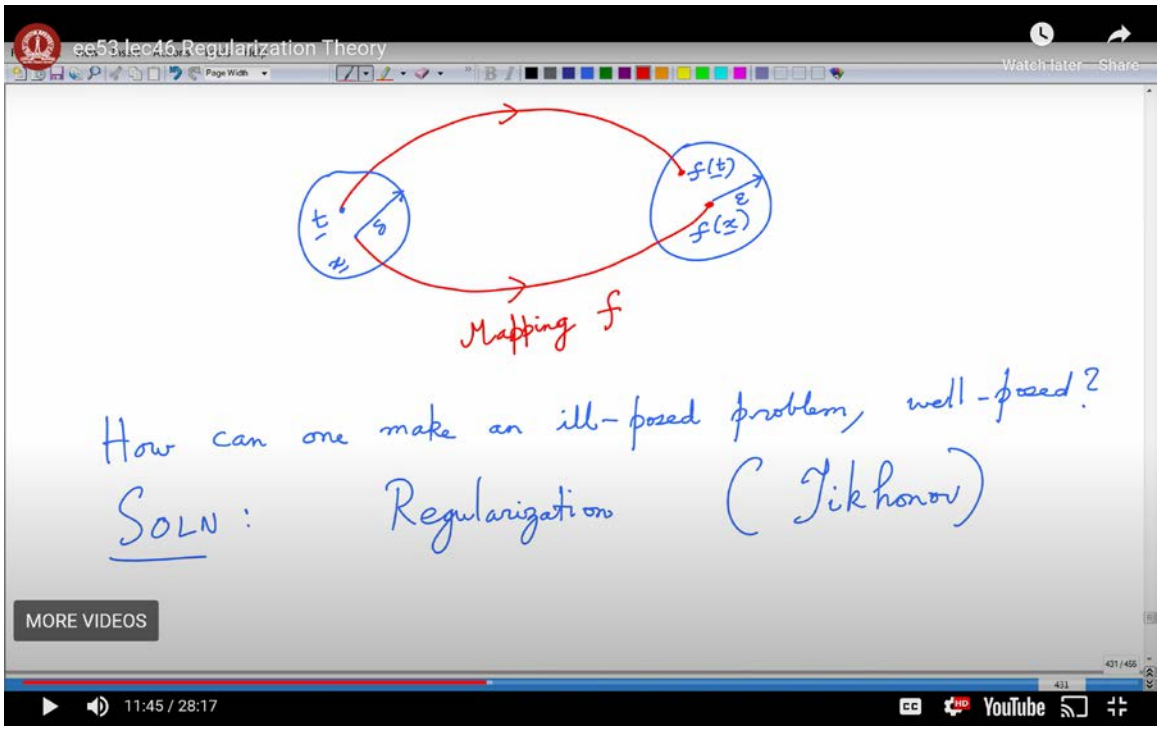
Now, for the reconstruction problem to be well-posed, it must satisfy three essential conditions: existence, uniqueness, and continuity.

1. Existence: For every input vector x in the domain X, there must exist a vector y = f(x) in the range Y. This ensures that every input x has a corresponding output y in the mapping.

2. Uniqueness: For any pair of input vectors x and t in the domain X, if f(x) = f(t), it must imply that x = t. This means that if two points in the range coincide, their corresponding points in the domain must be identical. Uniqueness is crucial because if the mapping is not unique, the association can be incorrect. For example, if you map an image of a cat to a label, and the mapping isn't unique, reversing the process (inverse mapping) could lead to ambiguity. The uniqueness ensures that if two domain vectors are the same, their mapped values in the range will also be the same.

3. Continuity: For any small positive value ϵ, there exists a corresponding δ, which is a function of ϵ, such that if the distance between x and t is less than δ, then the distance between f(x) and f(t) will be within ϵ. This concept can be visualized: imagine two points x and t in the domain, with t within a ball centered at x with a radius of δ. If this distance is less than δ, then their images under f, i.e., f(x) and f(t), will lie within an ϵ-radius ball in the range. This is the essence of the continuity constraint.

(Refer Slide Time: 11:45)



Uniqueness is particularly significant when dealing with optimization problems, as we aim to find a unique solution. If multiple solutions exist, it raises the question of which solution is the best, thus rendering the problem ill-posed.

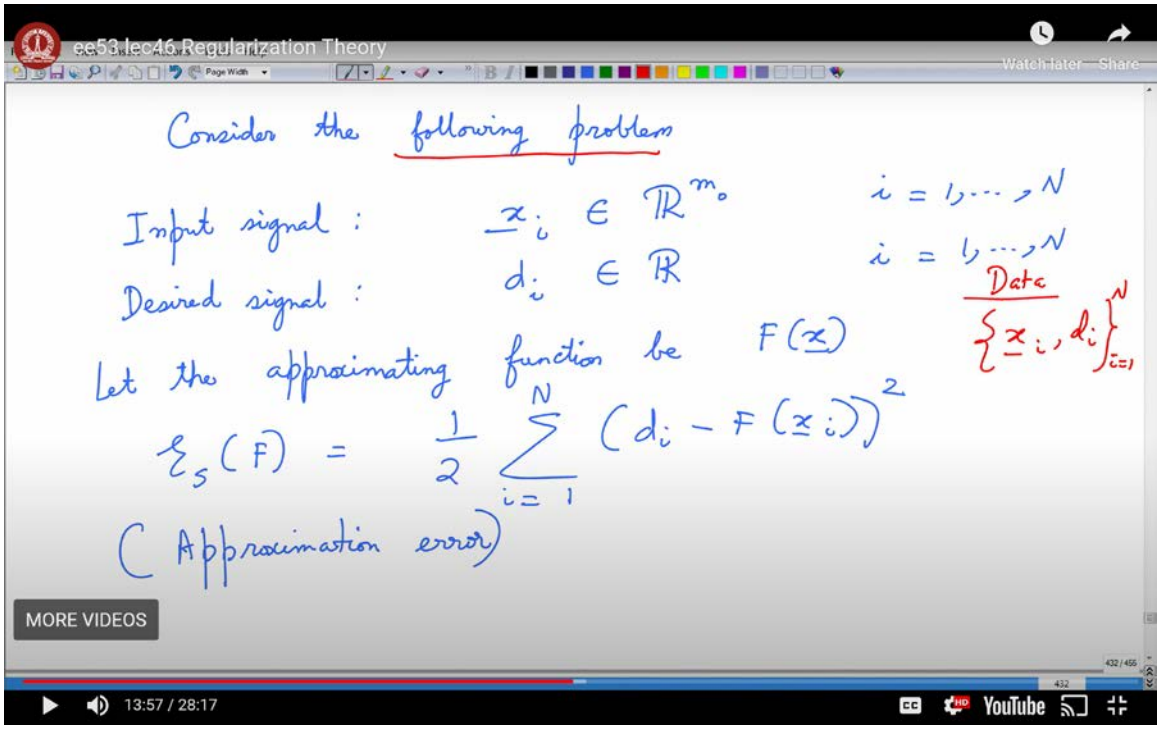
Thus, for a problem to be well-posed, it must satisfy the conditions of existence, uniqueness, and continuity. These criteria ensure that the solution to the problem is well-defined and reliable. However, when a problem is ill-posed, meaning it doesn't satisfy these conditions, we need to apply regularization to transform it into a well-posed problem. Regularization introduces additional constraints or terms to the functional, helping us to achieve a stable and unique solution.

Tikhonov proposed this theory long ago, establishing a foundation for making ill-posed problems well-posed by imposing specific constraints on the mapping, depending on the problem's geometry. Let's delve into how this can be set up in practice.

Consider the following scenario: we have an input signal consisting of data points x_i, where each x_i belongs to an m_0-dimensional space, indexed by i = 1 to n. Additionally, we have a

corresponding desired signal, denoted as d_i, which is associated with each x_i. For simplicity, let's assume that d_i is a scalar, although it could be a vector in a higher-dimensional space. Essentially, these pairs (x_i, d_i) constitute our dataset.

(Refer Slide Time: 13:57)



Our goal is to form an approximating function F, which allows us to calculate the approximation error. This error is essentially the squared difference between the desired signal d_i and the function's output $F(x_i)$ for each data point x_i. The desired signal d_i drives the learning process by guiding how F should behave to minimize this error.

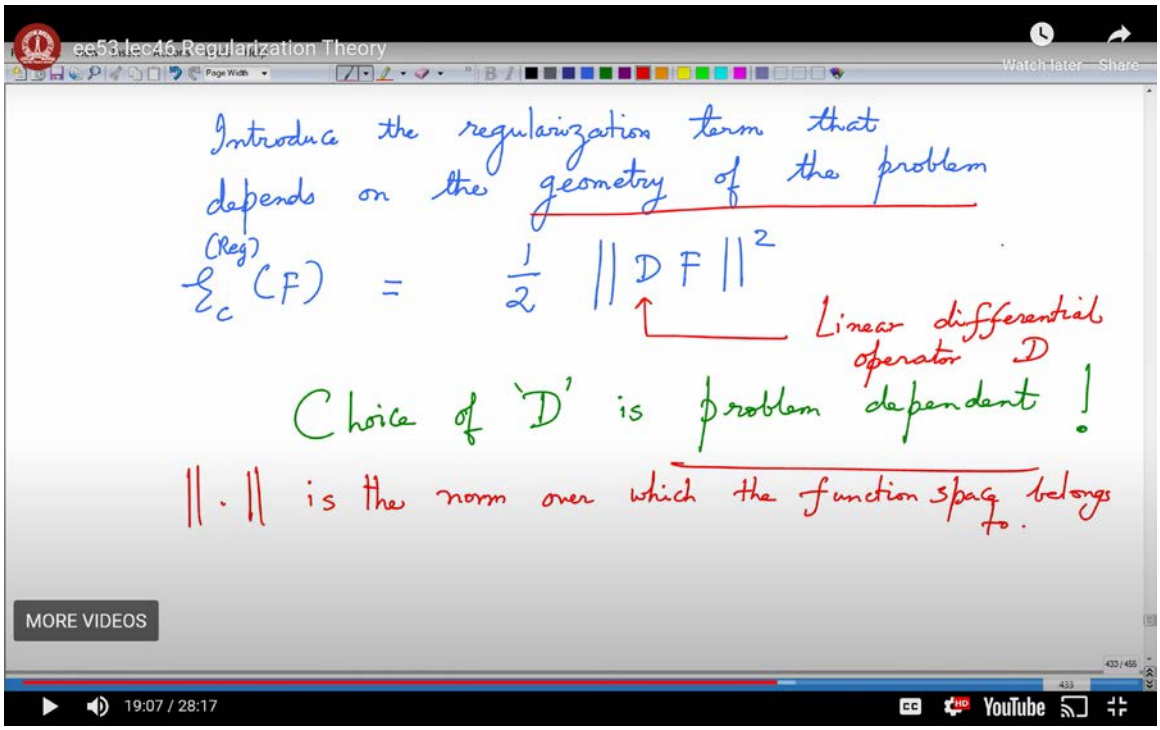
Now, while I haven't normalized the error by a scale factor N, we'll set that aside for now. Instead, we'll introduce a regularization term that depends on the problem's geometry. This is crucial because regularization helps impose necessary constraints, ensuring that the solution adheres to the problem's specific conditions.

Let's consider an example to clarify this. Imagine you're solving a path-planning problem where you need to travel from point A to point B within a city. The path you choose might aim to minimize your commute time, but in doing so, you may end up selecting a longer

route, such as a freeway, which could increase your fuel consumption. These are the types of constraints you must consider. By imposing such constraints, you regulate the possible paths and ultimately select the one that best solves your problem.

Another example involves determining the position of a particle moving from point A to point B. Here, you might impose constraints like limiting the particle's velocity or acceleration. These constraints influence how you vary the particle's speed and acceleration, allowing you to accurately determine its position.

(Refer Slide Time: 19:07)

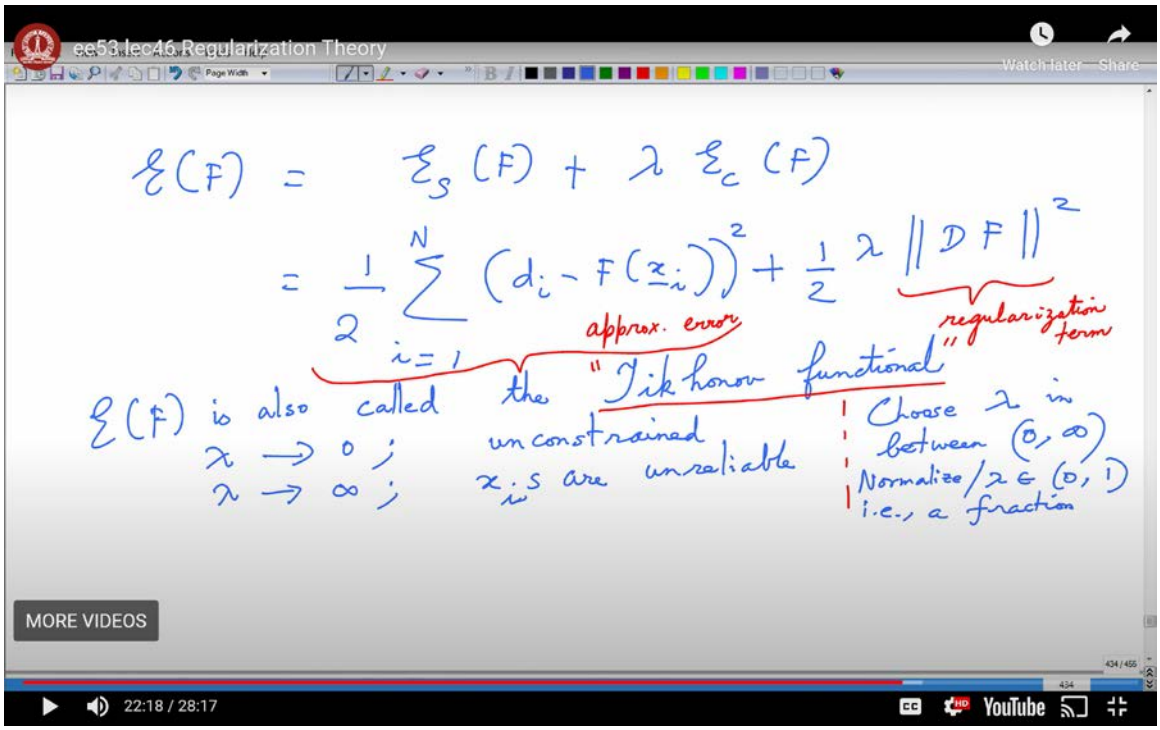


In the context of neural networks, a similar approach applies. Suppose you have a set of data points and want to impose constraints on the sparsity in the network's parameter space. For instance, you might aim to eliminate dead neurons or impose conditions on synaptic connections. These are regulatory constraints that guide the learning process, ensuring that the network's behavior aligns with the problem's geometry.

Regularization terms can be introduced into our functional to manage the constraints, and typically, this functional is expressed as one-half the norm of D × F squared, where D is a

linear differential operator acting on F. This operator can involve higher-order derivatives, first derivative, second derivative, and so on, up to the n-th derivative. Essentially, this linear differential operator is applied to the approximating function F within these defined constraints. The choice of this differential operator is highly dependent on the problem at hand. For example, I may restrict myself to the first derivative, or I may need to consider second derivatives, depending on the specific constraints that must align with the geometry of the problem.

(Refer Slide Time: 22:18)

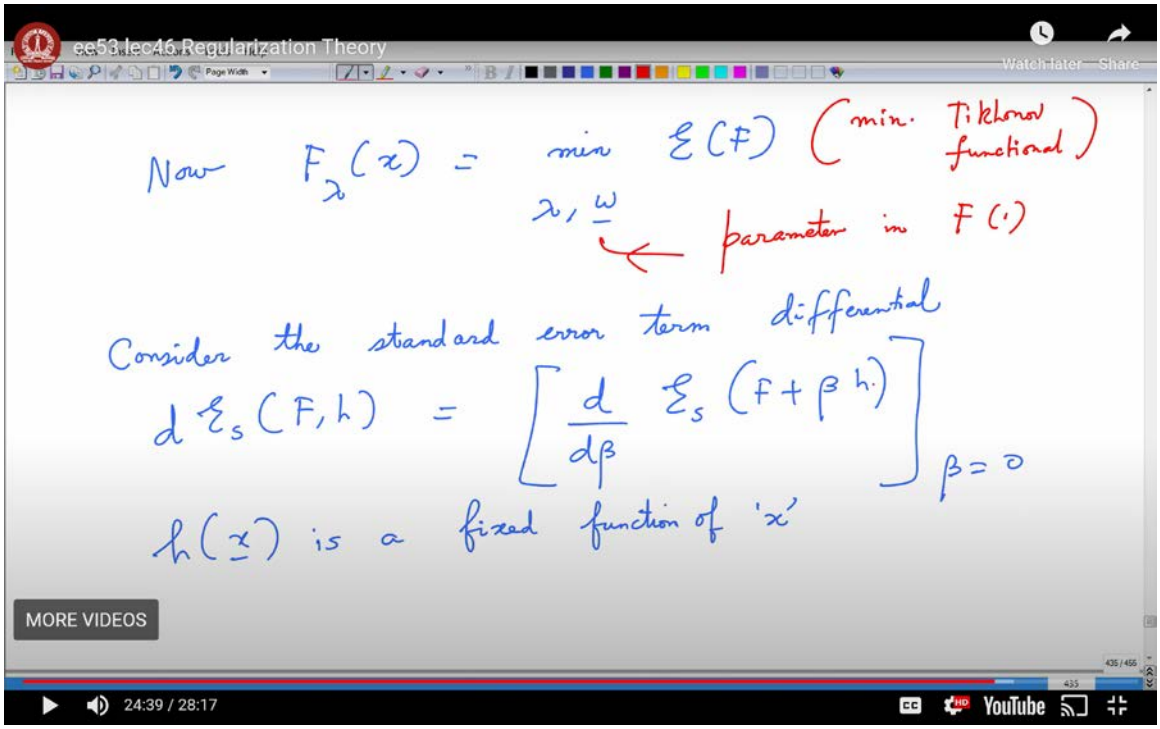


The norm in this context refers to the space in which the function F resides. Typically, we work with L_2 norms, where the geometry of the L_2 norm is a circle. You might be familiar with different norms: L_1 norm resembles a rhombus, L_2 norm forms a circle, and there are others like the sup norm, each with its distinct geometry. The selection of the appropriate norm depends on the problem, and it defines the geometry where this function resides.

For a more detailed exploration of norms and their geometries, I recommend referring to my other course, "Mathematical Methods and Techniques for Signal Processing," where these topics are discussed in-depth.

Now, let's break down the functional into two main components. The first component is $E_s(F)$, which corresponds to the approximation error, and the second component is $E_c(F)$, which is the regularization term. These two functionals, $E_s(F)$ and $E_c(F)$, are coupled by a parameter λ. The overall functional $E(F)$ is known as the Tikhonov functional, hence the term "Tikhonov regularization," because it involves introducing this regularization term into the functional to solve the hypersurface reconstruction problem subject to these regulatory conditions.

(Refer Slide Time: 24:39)

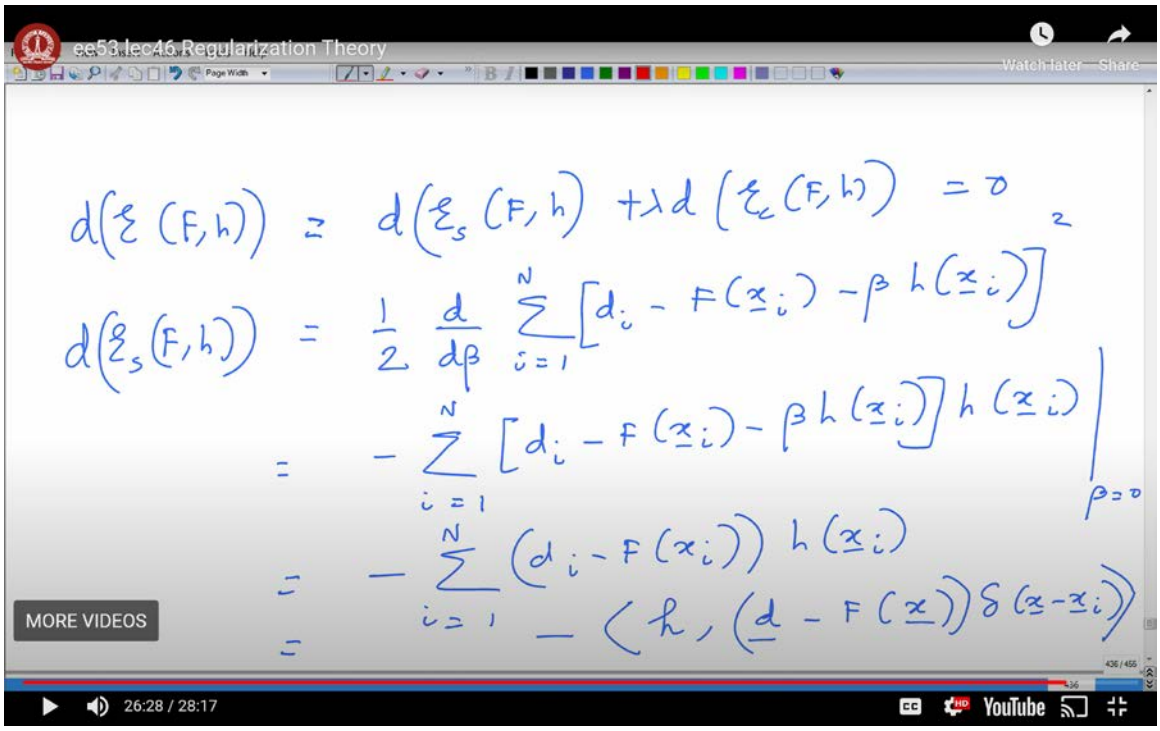


Now, how do we regulate? This regulation is controlled by the parameter λ. When $\lambda = 0$, the problem is entirely unconstrained, meaning we focus solely on the data points. On the other hand, when λ approaches infinity, the data points become irrelevant, and the

regularization term dominates, meaning the optimization is heavily influenced by the regularization.

In practical scenarios, λ is chosen between 0 and infinity, and typically, we normalize it so that λ falls between 0 and 1. Here, 0 represents the fully unconstrained case, while 1 indicates a scenario driven entirely by the regularization term.

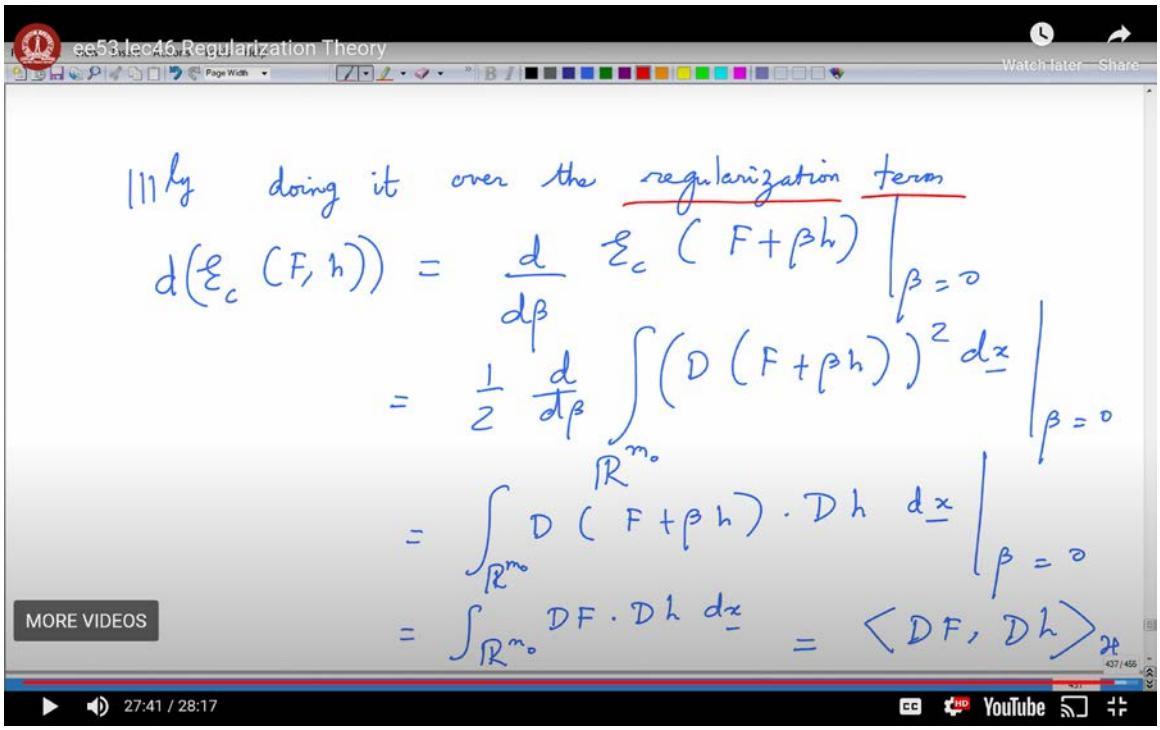
(Refer Slide Time: 26:28)



Returning to our problem, the goal is to find the approximating function $F_\lambda(x)$, which minimizes the Tikhonov functional. In other words, we are minimizing the Tikhonov functional over both λ and w. Here, λ is the regulatory parameter, and you might be wondering where w comes into play. w represents the parameters within this approximating function. Since you are familiar with radial basis functions, you know that the approximating function can be expressed as a linear combination of these radial basis functions, with the weights being the w parameters. Similarly, in the context of the backpropagation algorithm, w represents the hidden weights.

Next, let's delve into the exploration of the standard error term differentials in E_s and E_c by performing a small expansion around the vicinity of the functional to better understand their implications. Consider the standard error term, $E_s(f, h)$, and examine the differential $dE_s(f, h)$. The approach involves taking a small linear approximation, where we consider $f + \beta h$, with h being a fixed function of x. By taking the derivative and setting $\beta = 0$, we compute this differential.

(Refer Slide Time: 27:41)



Our goal is to set this differential of the overall Tikhonov functional to zero, aiming for an optimality criterion. This involves looking at the derivative and differential with respect to E_s and E_c, setting them equal to zero. However, when optimizing this functional, the optimization must be carried out over both the parameter λ and w. Invoking the optimality criterion here involves considering this small differential, which essentially equals zero.

To expand on this, I take the differential, expressed as,

$$\frac{1}{2}\frac{d}{d\beta}\sum_{i=1}^{n}\left(d_i - f(x_i) - \beta h(x_i)\right)^2.$$

Then, I perform a straightforward derivative with respect to β. The negative sign becomes apparent, and the square and half terms cancel out, leading to $\sum_{i=1}^{n}\big(d_i - f(x_i) - \beta h(x_i)\big)h(x_i)$, evaluated at β = 0. This expression can be viewed as an inner product between h(x) and d - f(x), evaluated at the points $x = x_i$.

Similarly, by performing the differential computation over the regularization term, we compute $\frac{d}{d\beta}E_c(f + \beta h)$ evaluated at β = 0. The result is essentially half of $\frac{d}{d\beta}[\int(Df + \beta Dh)^2\ dx]$, integrated over the entire m_0-dimensional space. Simplifying further, this leads to taking an inner product between Df and Dh in this space H, where the inner product is defined.

With this understanding, we are now prepared to embark on our journey with the Euler-Lagrange equation, which connects these differentials. This connection will allow us to further explore and derive meaningful results. With this, we conclude this module and will continue to build on these concepts in the next session.