Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani

Department of Electronic System Engineering

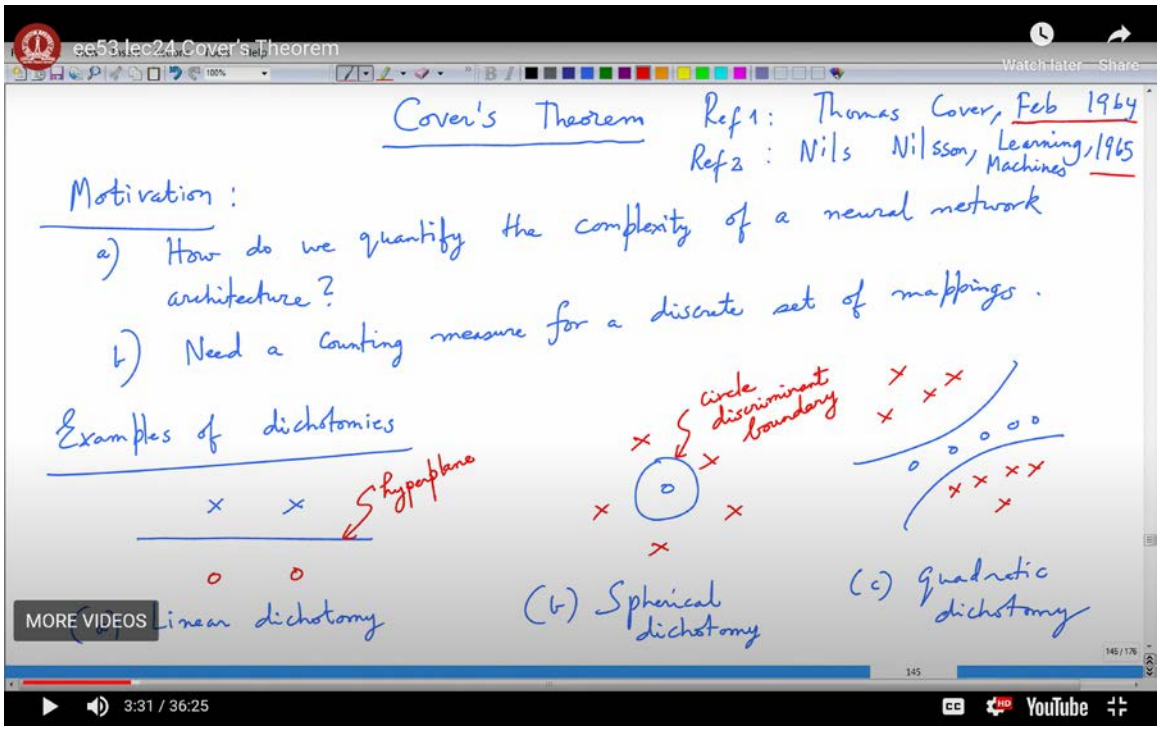
Indian Institute of Science, Bengaluru

Lecture – 24

Cover's Theorem

Welcome, everyone. Let's dive into Cover's Theorem. This topic is based on the work of Professor Thomas Cover, published in February 1964, and further elaborated in Nils Nilsson's 1965 book on learning machines. To start, we need to understand the motivation behind this theorem: quantifying the complexity of a neural network architecture, particularly for classification tasks.

(Refer Slide Time: 03:31)



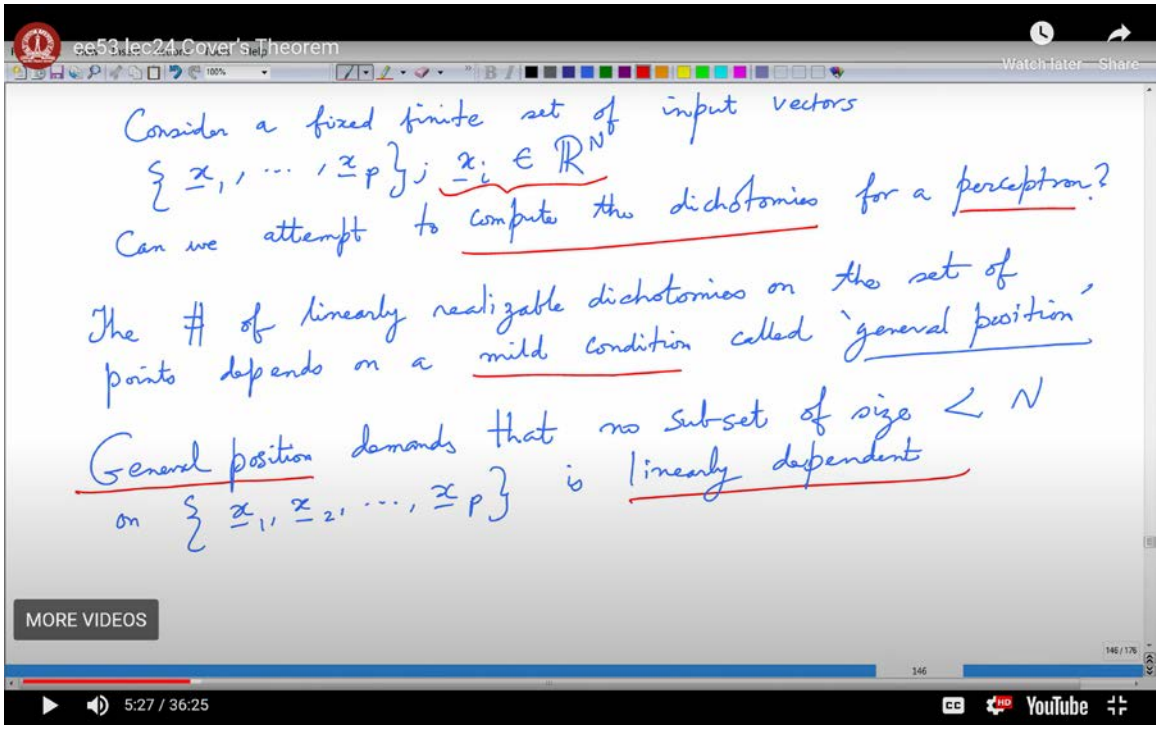
The core question is: How do we measure the complexity of a neural network when performing a classification task? To address this, we need a method to count the number

of discrete mappings, which serves as our motivation.

Consider the problem of ensuring linear separability. How can we quantify the complexity of achieving this? Let's look at a few examples of dichotomies to illustrate this concept.

In Figure A, we see a linear dichotomy where a hyperplane separates two sets of points, crosses and circles, because the discriminant boundary is a plane. But linear dichotomies are just one type. For instance, Figure B shows a spherical dichotomy, where the circles are surrounded by crosses. Here, the discriminant boundary is a circle, effectively separating the two patterns.

(Refer Slide Time: 05:27)

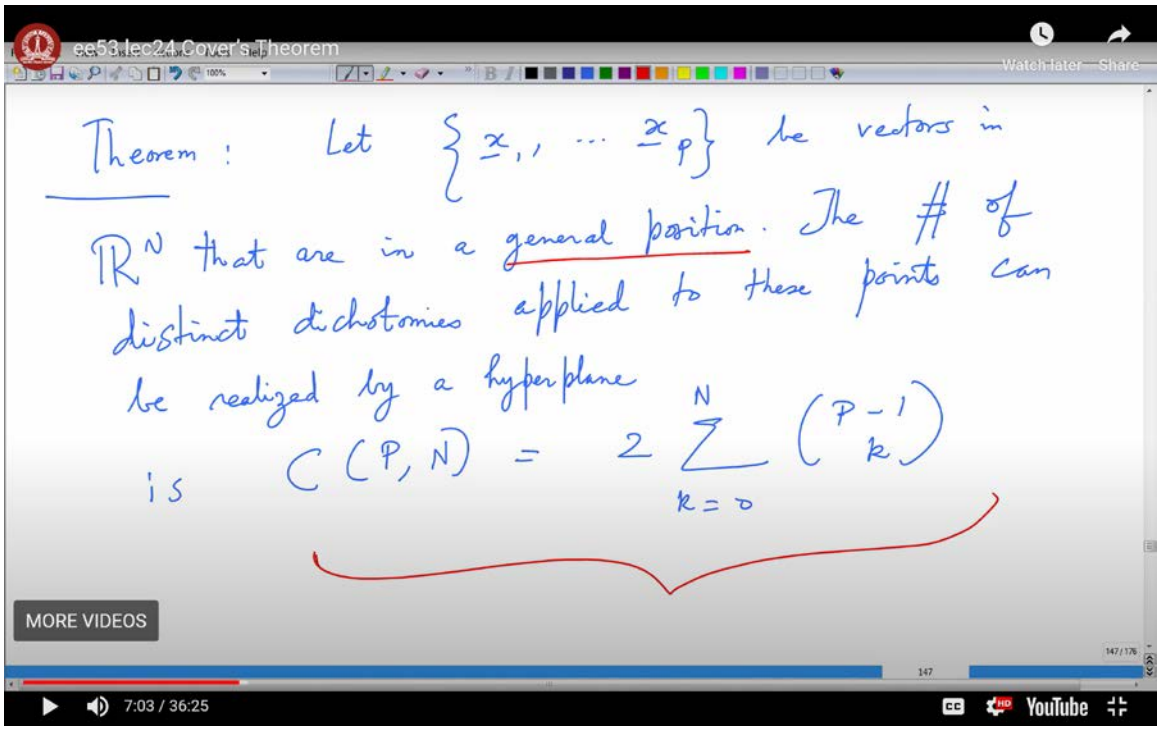


We can also have quadratic dichotomies, where conic sections like parabolas or ellipses serve as the discriminant boundary. This illustrates that the boundary can range from a simple hyperplane to more complex curves. Neural networks are capable of generating these complex discriminant boundaries, whereas simpler pattern recognition systems typically use linear boundaries.

Throughout this module, we'll focus on linear discriminant boundaries for simplicity.

Now, let's formulate the problem more precisely. Consider a fixed, finite set of input vectors x_1, x_2, …, x_p, where each vector x_i resides in an n-dimensional space. We will explore how to compute the dichotomies for a perceptron, the first algorithm we encountered, and determine how many distinct ways we can classify these points using the perceptron model.

(Refer Slide Time: 07:03)

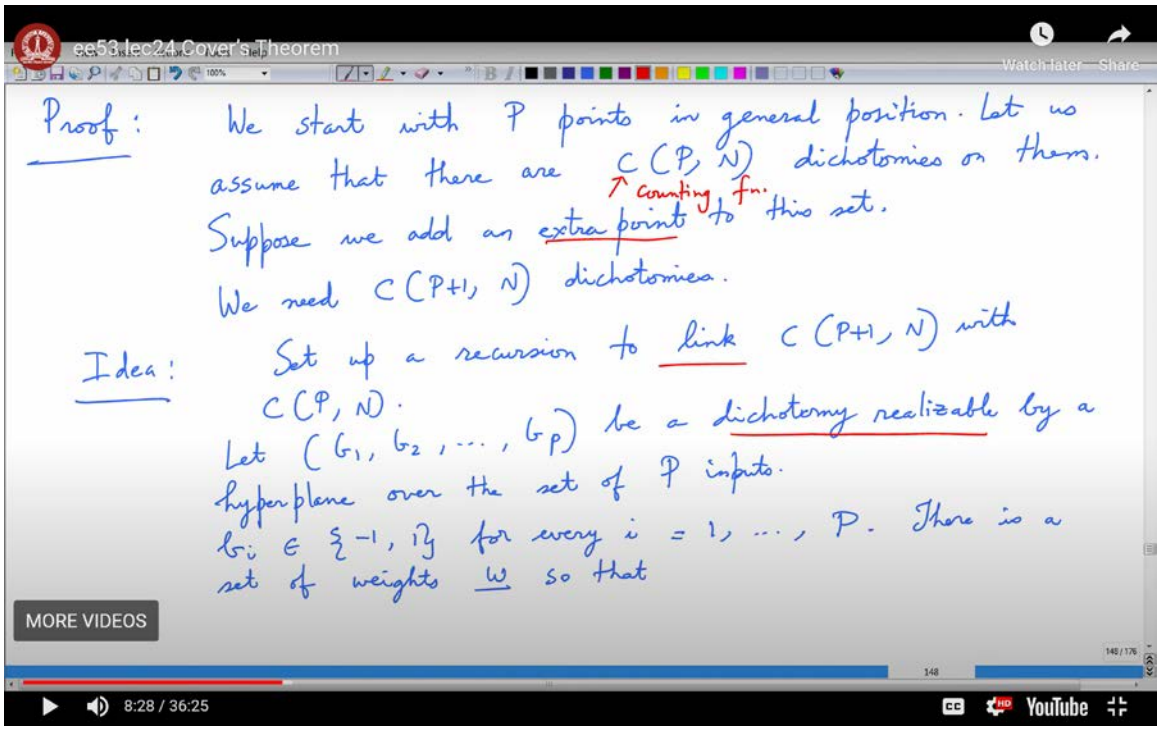


The number of linearly realizable dichotomies for a set of points depends on a condition known as "general position." To clarify, the number of linearly realizable dichotomies depends on this condition of general position. Specifically, general position requires that no subset of size less than n, where n is the dimension of the input vector, is linearly dependent. In other words, no subset of size smaller than n among these vectors should be linearly dependent. This is a key concept from linear algebra, concerning linear independence and dependence.

With this in mind, let's state the theorem. Consider a set of vectors $x_1, x_2, \ldots, x_p$ in R^n that

are in general position. Recall what general position entails: the number of distinct dichotomies that can be applied to these points, and which can be realized by a hyperplane, is given by a combinatorial quantity. This quantity is denoted as C(p, n), where p is the number of points and n is the dimension. It is expressed as $2 \times \sum_{k=0}^{n} \binom{p-1}{k}$.

(Refer Slide Time: 08:28)



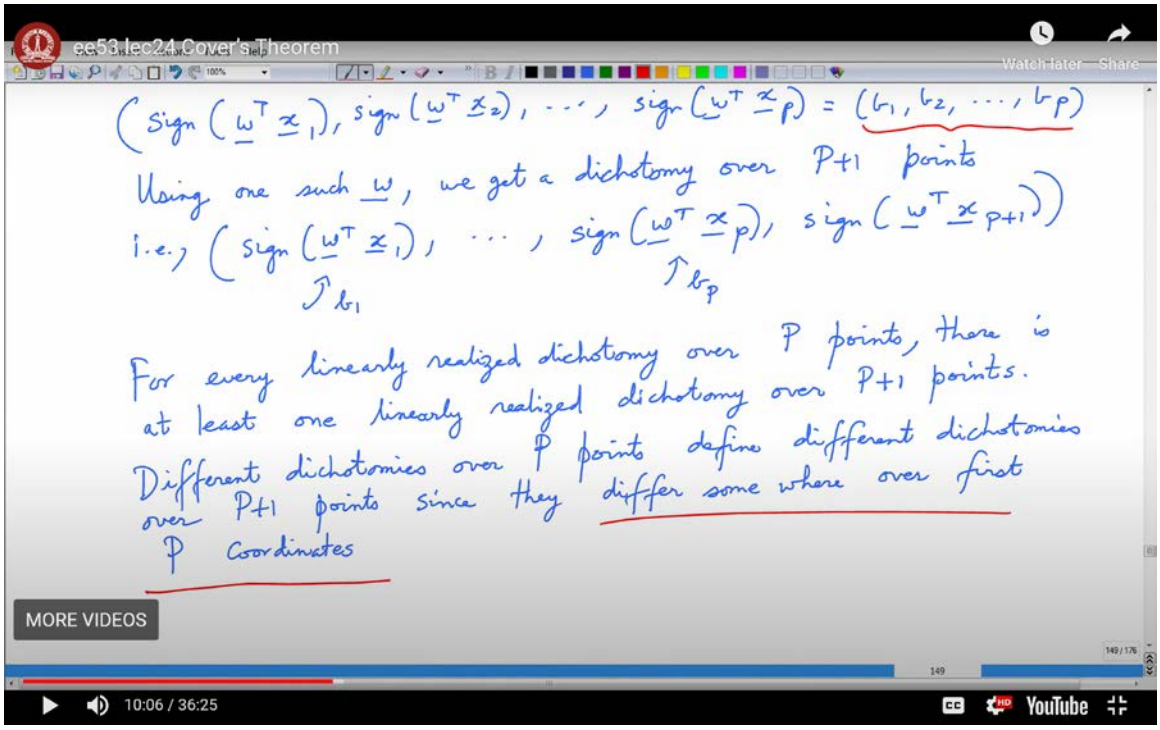
To approach the proof, one common method is to use induction, but it is crucial to establish the boundary conditions carefully to ensure that the result holds true.

Why is this result significant? It demonstrates that as we move to higher dimensions, the likelihood of having linearly separable patterns increases. In other words, the probability of achieving linear separability among a set of data points improves with increasing dimensionality.

Let's begin with the proof. Assume we have p points in general position, and let's denote the number of dichotomies as C(p, n). Now, if we add an additional point to this set, say x_{p+1}, we then have C(p+1, n) dichotomies.

The key idea is to establish a recursion that relates C(p+1, n) to C(p, n). Consider a dichotomy $b_1, b_2, \ldots, b_p$ that is realizable by a hyperplane over the set of p points. Here, each b_i is either -1 or 1 for each point x_i. There exists a set of weights w such that the sign of $w^T x_i$ yields b_i for each i.

(Refer Slide Time: 10:06)



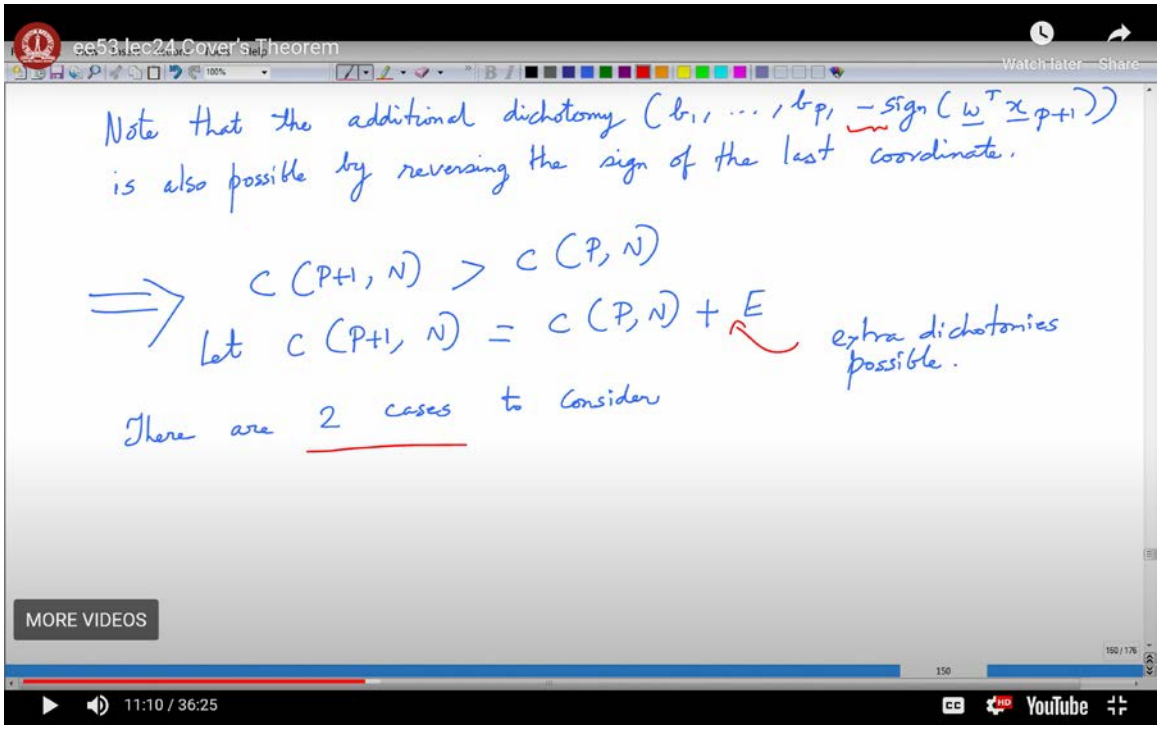
Consider a hyperplane represented by the weight vector w. When I compute the inner product of w with a given point and then apply the sign function, I obtain the attributes $b_1, b_2, \ldots, b_p$. By using this weight vector w, we achieve a dichotomy over p+1 points, denoted by this tuple. Specifically, for the new point x_{p+1}, I apply the same hyperplane, compute the inner product, and then take the sign to get b_{p+1}.

For every linearly realizable dichotomy with p points, there is at least one linearly realizable dichotomy with p+1 points. Different dichotomies for p points lead to distinct dichotomies for p+1 points, because they vary in the first p coordinates. This emphasizes the combinatorial aspect inherent in the geometry of this problem.

Note that the additional dichotomy is achieved by altering the sign of the last coordinate.

We can assign a plus or minus sign to the point x_{p+1}. This implies that C(p+1, n) is greater than C(p, n), because with p+1 points, we can have additional dichotomies. Let E be this extra term that satisfies the equation C(p+1, n) = C(p, n) + E, showing that equality holds with the additional term.

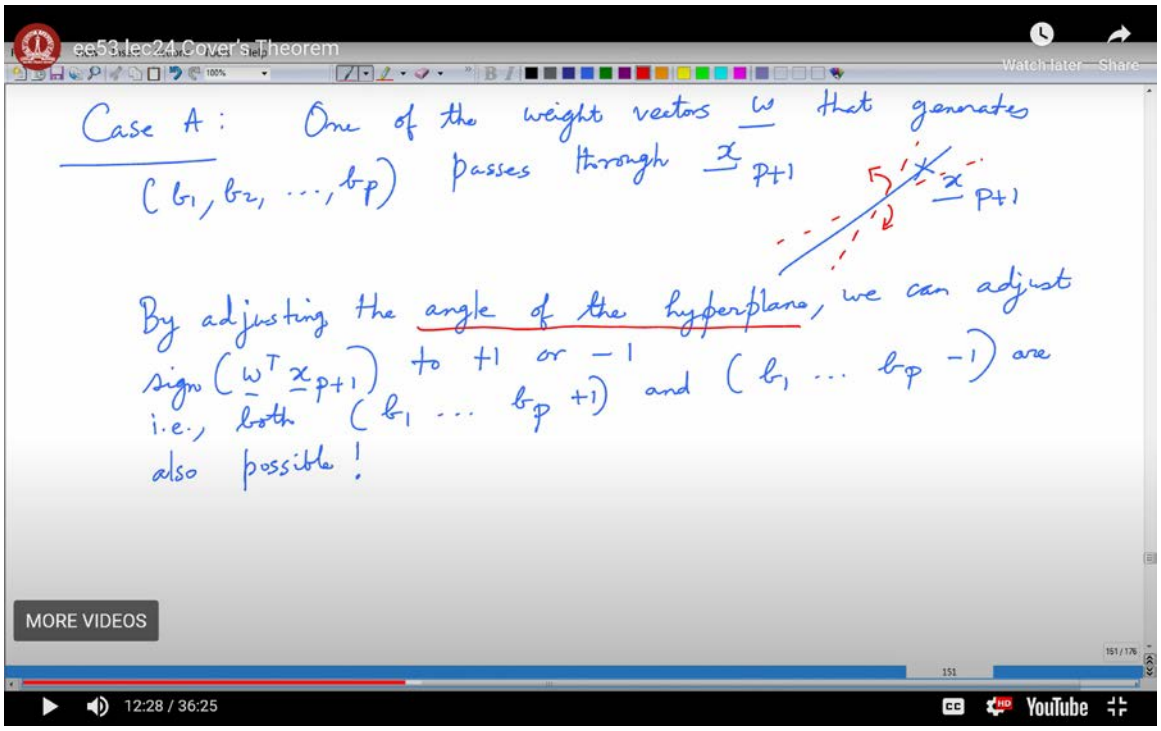
(Refer Slide Time: 11:10)



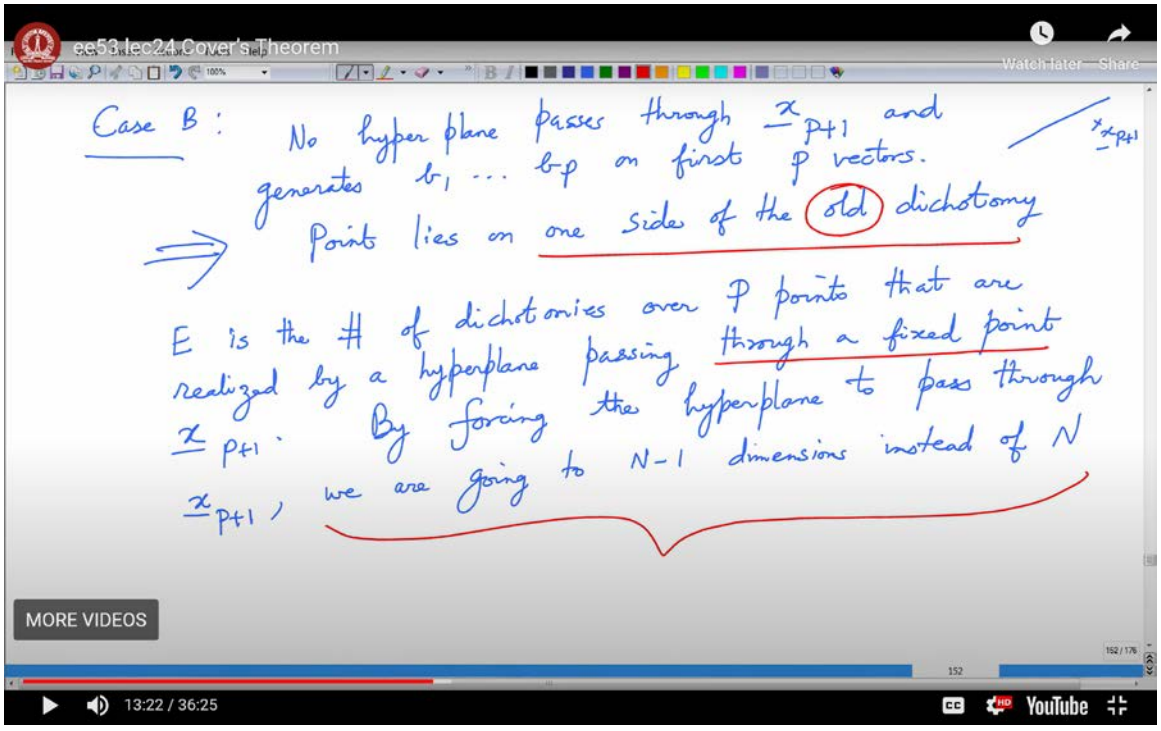
There are two cases to consider:

1. **Case A:** The weight vector w that generates the coordinates b_1, b_2, …, b_p passes through the point x_{p+1}. In this scenario, we need to adjust the sign. By varying the angle of the hyperplane, we can set $w^T x_{p+1}$ to be either +1 or -1. Tilting the hyperplane slightly in one direction gives one result, while tilting it in another direction gives a different result. Therefore, both possibilities for the last coordinate of the tuple b_1, b_2, …, b_p with respect to x_{p+1} being either +1 or -1 are feasible.

(Refer Slide Time: 12:28)



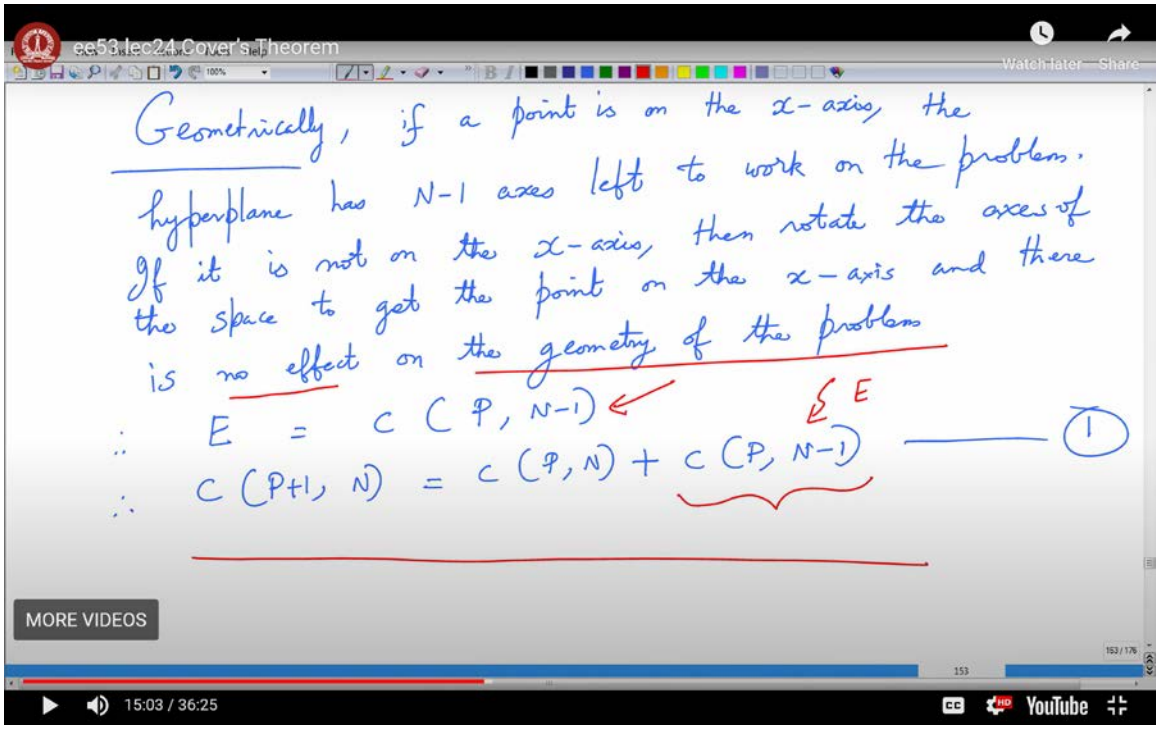
(Refer Slide Time: 13:22)



2. **Case B:** The hyperplane does not pass through x_{p+1}. In this case, the point x_{p+1} lies on one side of the existing hyperplane, making it clear which side it belongs to in the context of the old dichotomy.

Let E represent the number of dichotomies over p points that are realized by a hyperplane passing through a fixed point x_{p+1}. By constraining the hyperplane to pass through this point x_{p+1}, we effectively reduce the problem to n-1 dimensions instead of n dimensions. This is a crucial point to understand. For instance, if I have 10 points all lying on the x-axis, the dimension remains 1. This is the essence of the idea: geometrically, if a point lies on the x-axis, the hyperplane has n-1 dimensions left to operate within. If the point does not lie on the x-axis, we can rotate the coordinate system to position the point on the x-axis without affecting the problem's geometry.

(Refer Slide Time: 15:03)



Therefore, the number of additional dichotomies corresponds to the counting function C(p, n-1), since we are now working in n-1 dimensions. This leads us to the recursion formula:

$$C(p+1,n) = C(p,n) + C(p,n-1)$$

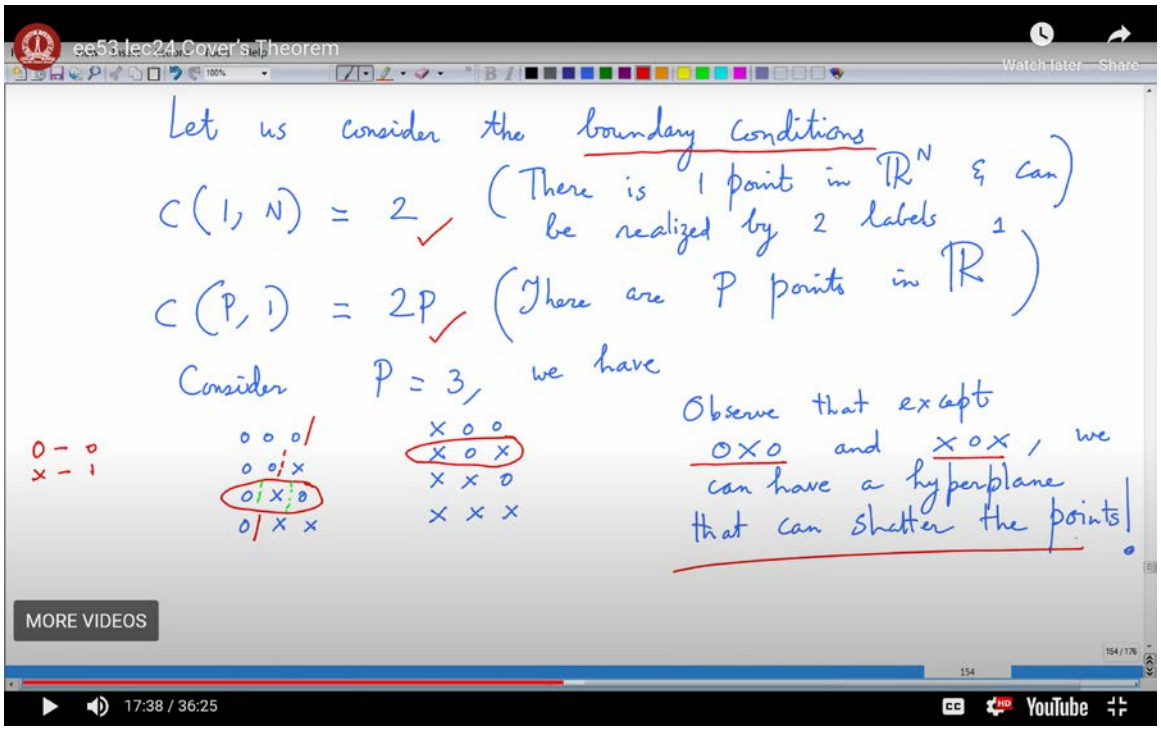
Here, C(p, n) is the number of dichotomies for p points in n dimensions, and C(p, n-1) accounts for the additional dichotomies when moving to n-1 dimensions.

To solve this recursion, we need boundary conditions. It's essential to establish these conditions carefully to ensure the validity of the proof. Inappropriate use of boundary conditions could lead to incorrect results. Let's consider the boundary conditions, following Nielsen's work:

1. First Boundary Condition: C(1, n) = 2. This indicates that with one point in R^n, there are two possible dichotomies: +1 and -1. This condition is quite straightforward.

2. Second Boundary Condition: C(p, 1) = 2^p. This means that with p points in R^1 (one dimension), the number of possible dichotomies is 2^p. This is a significant boundary condition, which can be demonstrated with an example.

(Refer Slide Time: 17:38)



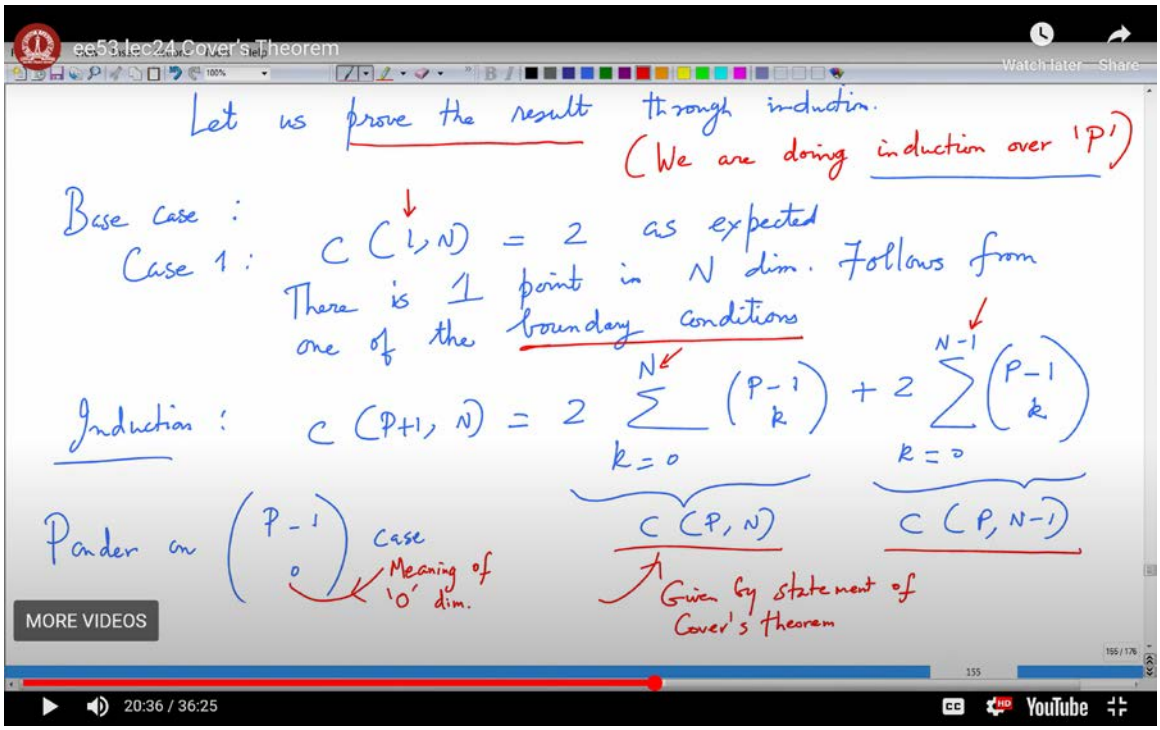
For instance, let p = 3. Suppose we have three points: circles and crosses. We need to

enumerate all possible patterns of dichotomies. If we denote circles as 0 and crosses as 1, we can map these patterns as Boolean combinations from 000 to 111. By placing a hyperplane in various positions, we can separate the circles and crosses according to each dichotomy pattern.

Thus, if we examine all possible configurations, we can verify that the number of dichotomies corresponds to 2^3, as expected. For each pattern, we can place a hyperplane that separates the circles from the crosses.

Let's explore a bit deeper. With a single hyperplane, I can't achieve the desired separation in some cases. For example, to separate certain patterns, such as one circle and two crosses, I would need two hyperplanes. This can be illustrated more clearly with different colors. For the more complex patterns, such as the combination of one circle and multiple crosses, a single hyperplane might not suffice, as it cannot realize the necessary dichotomy.

(Refer Slide Time: 20:36)



However, for simpler cases like separating one circle from two crosses, a hyperplane can

indeed be used to accomplish this separation. This highlights a critical boundary condition, which we will use to validate the proof of the theorem.

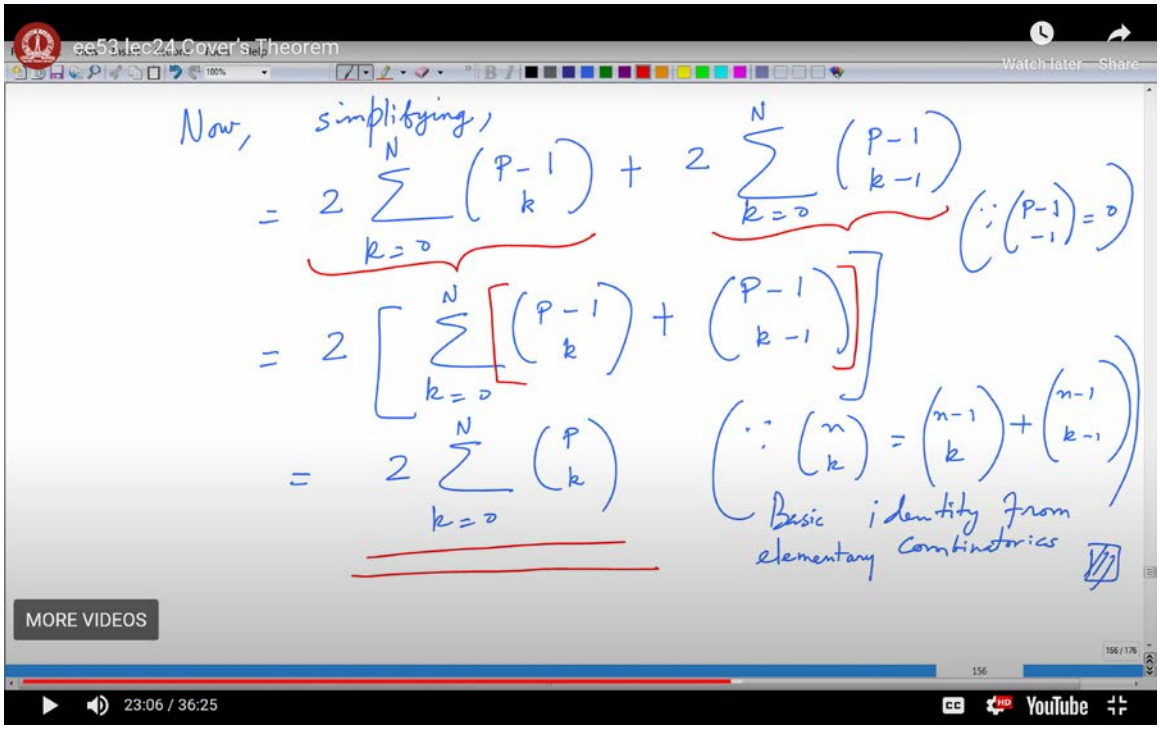
Now, let's prove the result using induction. We will begin by examining our steps. The base case involves C(1, n), where we have only one point and n dimensions. As expected, C(1, n) = 2, consistent with the boundary condition we discussed earlier.

The induction step proceeds by assuming that the result holds for p points in n dimensions. We need to prove it for p+1 points. Thus, the induction focuses on the number of points rather than the dimensions.

So, the recursion formula is given by:

$$C(p+1,n) = C(p,n) + C(p,n-1)$$

(Refer Slide Time: 23:06)



Expanding this, we have:

$$C(p,n) = 2 \times \sum_{k=0}^{n} \binom{p-1}{k}$$

and

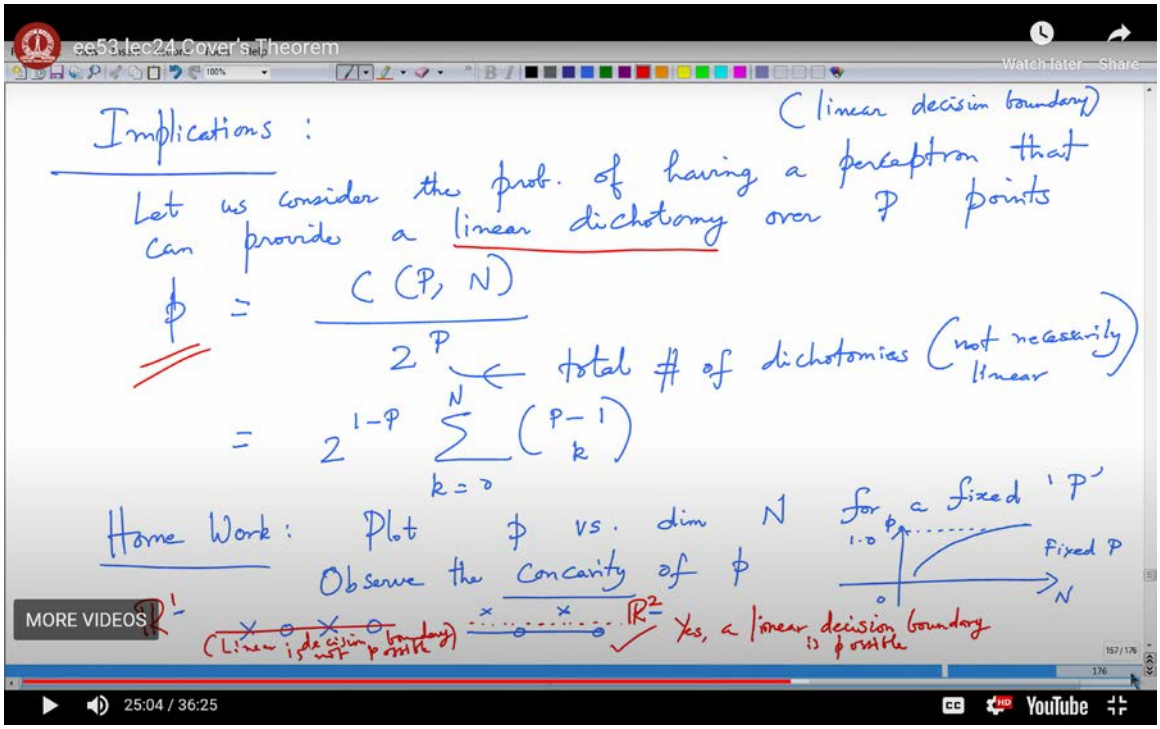
$$C(p,n-1) = 2 \times \sum_{k=0}^{n-1} \binom{p-1}{k}$$

Carefully observe the upper limit in the summation. When k = 0, the term $\binom{p-1}{0}$ is 1, which corresponds to the base case where p-1 is 0. This highlights why we have the factor of 2 in our result.

Next, we simplify the expression:

$$2 \times \sum_{k=0}^{n} \binom{p-1}{k} + 2 \times \sum_{k=0}^{n-1} \binom{p-1}{k-1}$$

(Refer Slide Time: 25:04)



In the previous slide, we had the upper limit as n-1. To match this with the new upper limit n, we adjust the summation term $\binom{p-1}{k-1}$, making the summation go from k = 0 to n.

By combining the two sums, we get:

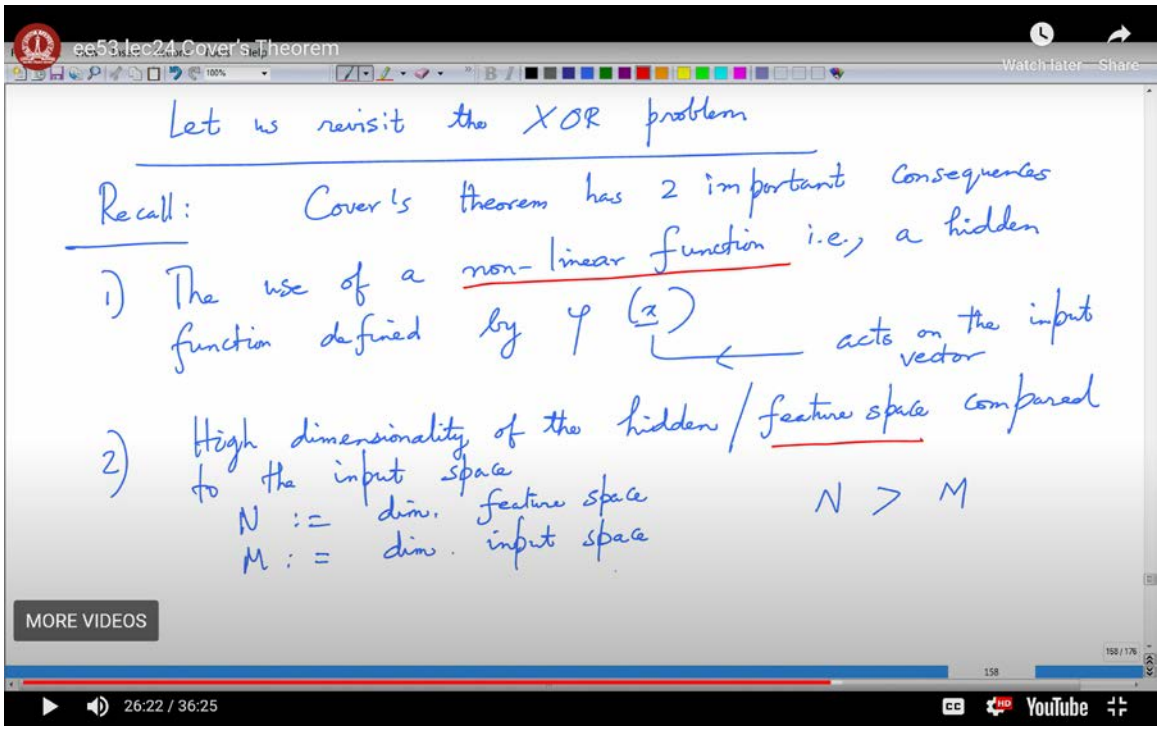
$$2\left(\sum_{k=0}^{n}\binom{p-1}{k}+\sum_{k=0}^{n}\binom{p-1}{k-1}\right)$$

Applying the fundamental combinatorial identity:

$$\binom{n}{k}=\binom{n-1}{k}+\binom{n-1}{k-1}$$

we get, $2\sum_{k=0}^{n}\binom{p}{k}$. This confirms our result, demonstrating the validity of the recursion formula.

(Refer Slide Time: 26:22)



A crucial point to emphasize is that if you assume the boundary conditions incorrectly, you might end up with a different expression where the summation ranges from k = 0 to n

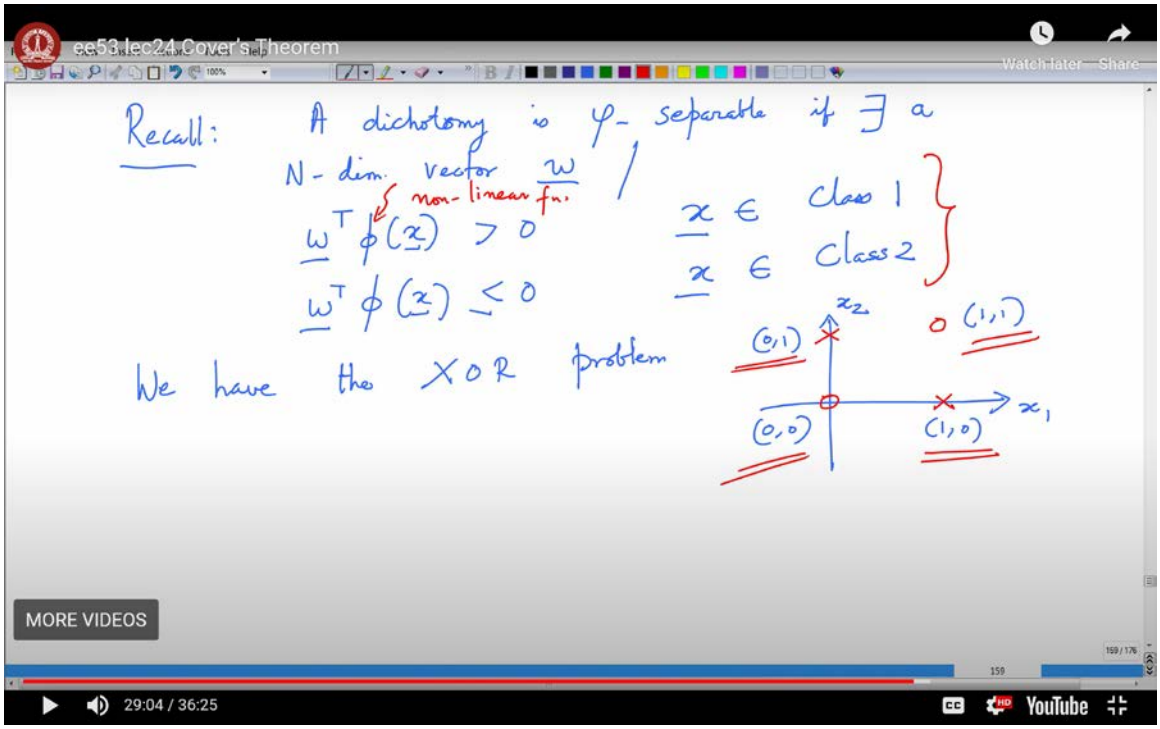
- 1. This could lead to slightly inaccurate results when enumerating dichotomies. However, if you adhere to the correct boundary conditions, the results will be valid, and induction will accurately verify the theorem. Therefore, it's essential to carefully consider the boundary conditions.

Now, let's explore the implications of this theorem. We are interested in determining the probability that a perceptron can provide a linear dichotomy over p points. This probability is given by:

$$\frac{C(p,n)}{2^p}$$

where C(p, n) represents the number of linear dichotomies possible with p points in n dimensions, and 2^p denotes all possible dichotomies.

(Refer Slide Time: 29:04)



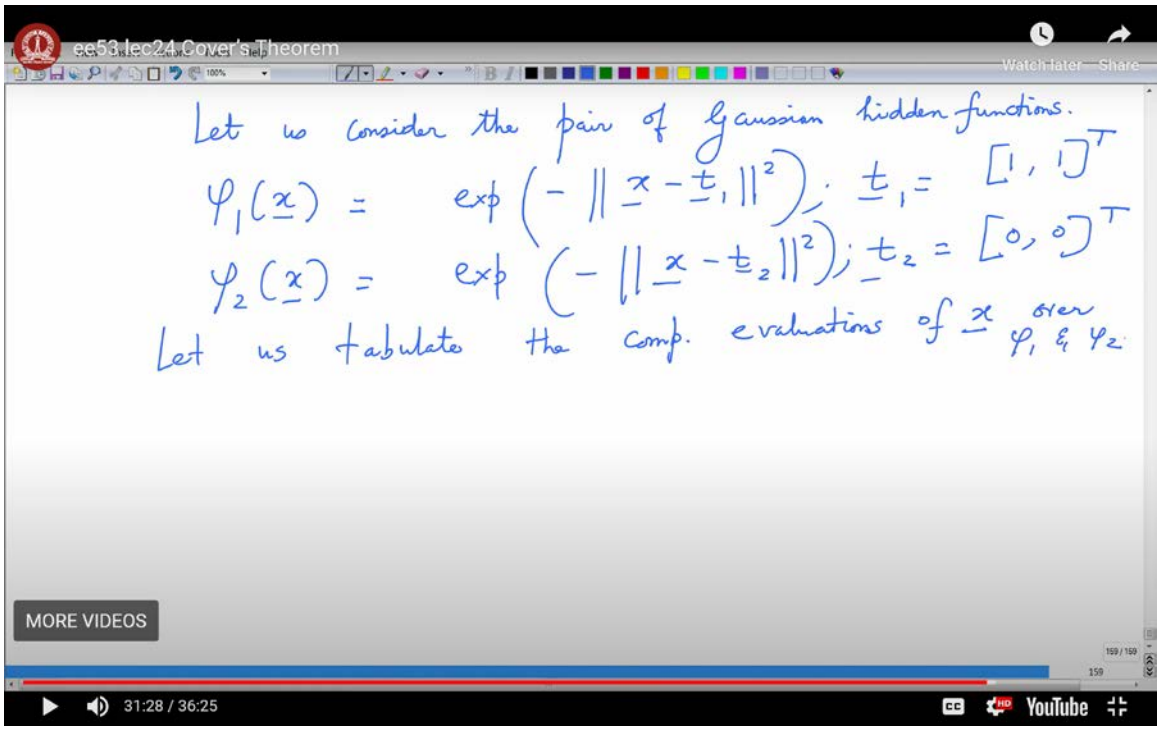
Since C(p, n) is less than 2^p, we can simplify the expression for this probability:

$$\frac{C(p,n)}{2^p} = \frac{2 \times \sum_{k=0}^{n} \binom{p-1}{k}}{2^p} = 2^{1-p} \times \sum_{k=0}^{n} \binom{p-1}{k}$$

I encourage you to plot this probability as a function of the dimension n for a fixed size p, which represents the number of points. Observe the concavity in the behavior of this plot. This is a significant implication, as it suggests that increasing the dimension improves the likelihood of finding a linear dichotomy for the given points, which enhances the potential for pattern separability.

Let's revisit the XOR problem. Although there's no direct implication from Coover's theorem to XOR, the underlying idea leads to an important consequence. We can utilize a non-linear function, often referred to as a hidden function, to map the input vector to a higher-dimensional space. In this feature space, the dimension n is greater than the original input dimension m, which may be slightly different from the notation used in Coover's theorem.

(Refer Slide Time: 31:28)



This consequence is crucial because by lifting data points to higher dimensions, we can

ensure linear separability. Why is linear separability important? It simplifies the problem, allowing us to use a hyperplane rather than a more complex neural network with a non-linear discriminant boundary.

To recall, a dichotomy is said to be φ-separable if there exists an n-dimensional vector w such that the inner product of w with $\varphi(x)$ is greater than 0 for all $\boldsymbol{x}$ belonging to class 1, and less than or equal to 0 for all $\boldsymbol{x}$ in class 2.

In the perceptron model, initially, we only had the condition that $\boldsymbol{w}^{\mathrm{T}}\,\boldsymbol{x} > 0$ for class 1 and $\boldsymbol{w}^{\mathrm{T}}\,\boldsymbol{x} \leq 0$ for class 2. This approach does not include a non-linear function φ. However, by introducing a non-linear function φ, we can achieve a dichotomy that is φ-separable, because we are now working in a transformed feature space.

Let's examine the XOR problem in this context. Consider a two-variable scenario with Boolean variables x_1 and x_2, which can be either 0 or 1. This gives us four possible combinations: (0,0), (0,1), (1,0), and (1,1). We will mark these coordinates as follows:

- Circles represent the coordinates (0,0) and (1,1).
- Crosses represent (1,0) and (0,1).

Clearly, the XOR problem is not linearly separable in its original form, as no single hyperplane can separate these points into two distinct classes.

Now, let's consider using Gaussian hidden functions to address this issue. In this context, a hidden space is also referred to as a feature space, and the terms hidden function, feature space, hidden map, and feature map are used interchangeably.

Let $\varphi_1(\boldsymbol{x})$ be defined as:

$$\phi_1(x) = \exp(-|x - t_1|^2)$$

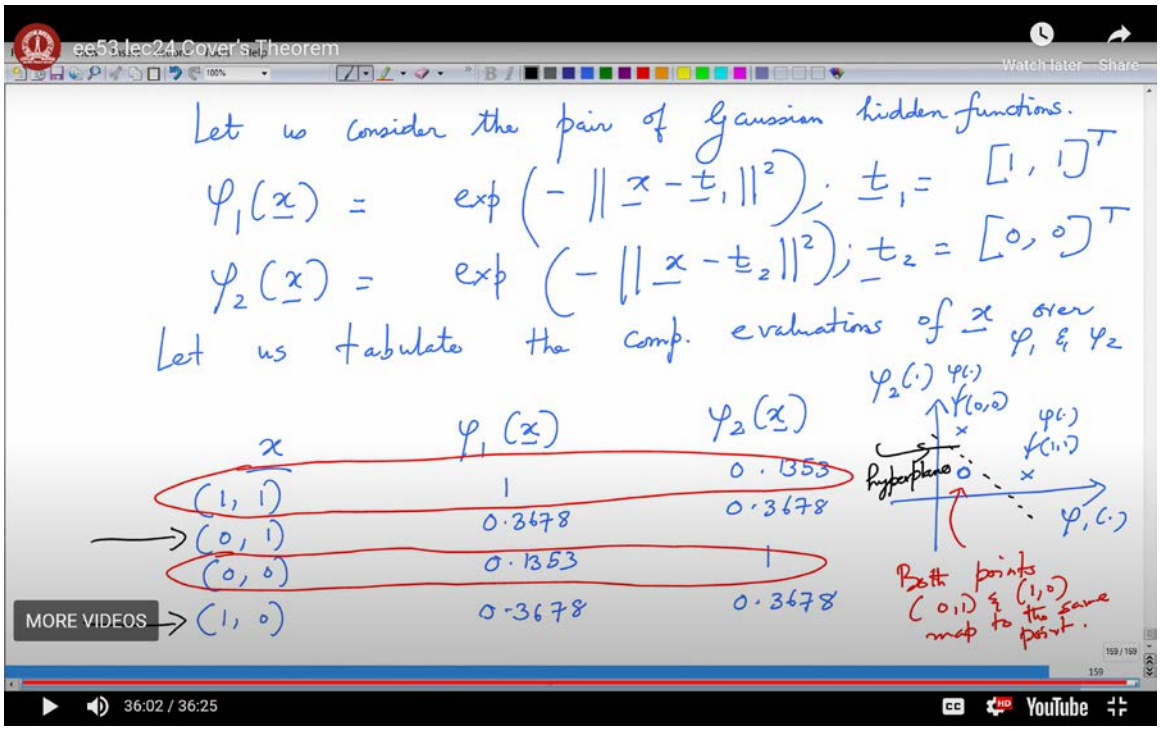
where $t_1 = (1,1)^T$ and x is a 2-dimensional vector.

Similarly, let $\phi_2(x)$ be defined as:

$$\phi_2(x) = \exp(-|x - t_2|^2)$$

where $t_2 = (0,0)^T$.

(Refer Slide Time: 36:02)



We will now compute $\varphi_1(\boldsymbol{x})$ and $\varphi_2(\boldsymbol{x})$ for the four points (1,1), (0,1), (0,0), and (1,0). The calculations are as follows:

- For (1,1):

$$\phi_1(1,1) = \exp(-|(1,1) - (1,1)|^2) = \exp(0) = 1$$

$$\phi_2(1,1) = \exp(-|(1,1) - (0,0)|^2) = \exp(-2) \approx 0.1353$$

- For (0,1):

$$\phi_1(0,1) = \exp(-|(0,1) - (1,1)|^2) = \exp(-1) \approx 0.3679$$

$$\phi_2(0,1) = \exp(-|(0,1) - (0,0)|^2) = \exp(-1) \approx 0.3679$$

- For (0,0):

$$\phi_1(0,0) = \exp(-|(0,0) - (1,1)|^2) = \exp(-2) \approx 0.1353$$

$$\phi_2(0,0) = \exp(-|(0,0) - (0,0)|^2) = \exp(0) = 1$$

- For (1,0):

$$\phi_1(1,0) = \exp(-|(1,0) - (1,1)|^2) = \exp(-1) \approx 0.3679$$

$$\phi_2(1,0) = \exp(-|(1,0) - (0,0)|^2) = \exp(-2) \approx 0.1353$$

Let's plot these results in the φ_1-φ_2 plane. The points (1,1) and (0,0) map to distinct coordinates, (1, 0.1353) and (0.1353, 1), respectively. The points (0,1) and (1,0) map to the same coordinate, (0.3679, 0.3679). This transformation enables us to use a linear boundary in the φ-space to separate these points.

Thus, by employing non-linear functions like Gaussian hidden functions, we can transform the original data into a higher-dimensional feature space where the points become linearly separable. This demonstrates that using hidden functions allows us to find a linear decision boundary in a transformed space, effectively solving the XOR problem.

This concludes our examination of the XOR problem with Gaussian hidden functions. We have successfully used a hyperplane in the feature space to separate the points. We'll pause here and continue with further topics in the next discussion.