

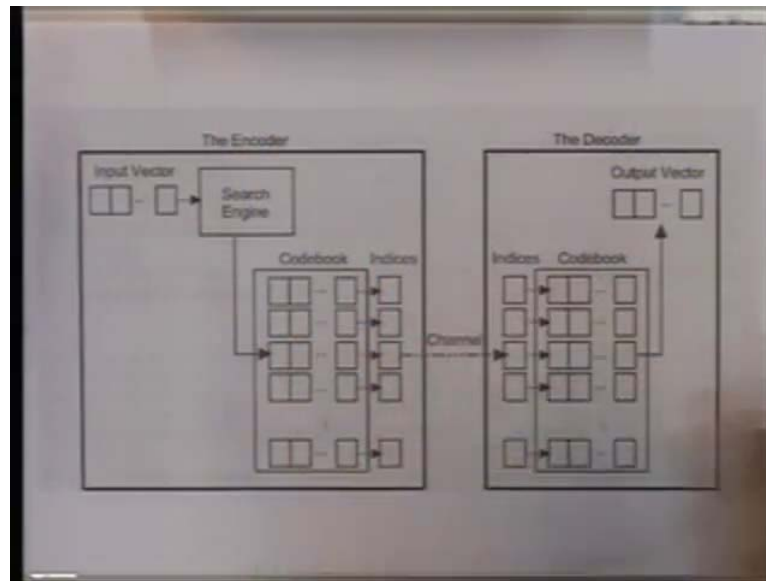
Information Theory and Coding
Prof. S. N. Merchant
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 39
Vector Quantization

We have studied different schemes for quantization. In all these schemes, the source outputs are quantized individually, that is they are treated as scalar values. And therefore, all these schemes fall under the category of what is known as scalar quantization. Earlier during our study on source encoding we had shown that by grouping the source outputs in blocks, and encoding them as a single block we could design efficient encoder in the sense that the rate of the encoder approach, the lower bound provided by the entropy of the source. Now, in a similar fashion we can extend this strategy to the case of quantization.

We can group the source output together and then quantize them, in a single block by doing. So, the performance of such a block quantizer will be better than the performance of a scalar quantizer, in the sense that given the rate the distortion obtained from such a block quantizer would be lesser than the distortion obtained from the equivalent scalar quantizer or given the distortion the rate for such a block quantizer will be lower than that obtained from the equivalent scalar quantizer. Now, this technique of block quantization is popularly known as vector quantization the figure here depicts.

(Refer Slide Time: 03:27)



The operation of a vector quantizer, a vector quantizer is composed of two operations the first is the encoder and the second is the decoder. The first step in vector quantization is to group the source output into blocks of vectors. For example, we can treat k consecutive samples of speech as the components of an k dimensional vector, this vector of source outputs form the input to the vector quantizer, at both the encoder and decoder of the vector quantizer, we have a set of k dimensional vectors. This set is called the code book of the vector quantizer, and the vectors in this code book are named as code vectors or code words.

And this code words of code vectors are selected to be representative of the vectors, we generate from the source output, each code vector or code word is assigned a binary index. The next step in vector quantization is performed by the search engine, the task of the search engine is to take an input vector, and compare it to each code vector in order to find the code vector, which is closest to the input vector. In this case the closeness is found by evaluating the Euclidean distance between the input vector, and each code vector or code word of the code book.

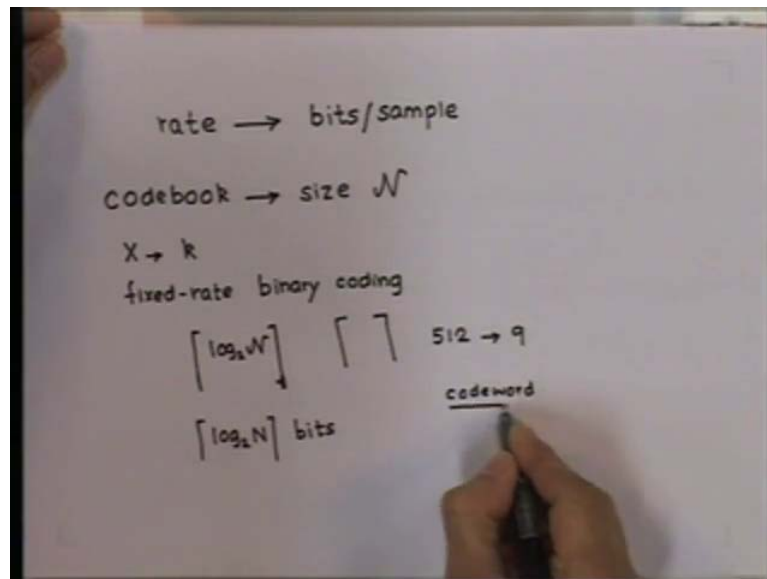
The elements of this code word are the quantized values of the source input. Once, the closest code word is found the index of that code word is sent through a channel, the channel could be a computer storage or a communication channel and so on. When the encoder receives the index of the code word and because the decoder has the exactly

code book as that exists in the encoder, it replaces the index with the associated code word.

In a vector quantizer the encoder has to perform of considerable amount of computation in order to find the closest code word to the vector of source outputs, but the decoder is very simple in the form of a table lookup. Therefore, vector quantization is a very attractive encoding scheme for applications, in which the resources available for decoding are considerably less than the resources available for encoding.

Now, there are two important issues related to vector quantization and this are how is the code book designed and how does the search engine work. Before we make an attempt to answer this question let us briefly discuss, the terminologies associated with vector quantization. As usual the amount of compression will be described in terms of the rate.

(Refer Slide Time: 08:11)



And this rate is measured in bits per sample, let us assume that we have a code book of size N , which implies that there are N code vectors of code words. Let their input vector be of dimension k , let us also assume fixed rate binary coding. Now, in order to inform the decoder of which code word was selected, we need to use \log to the base 2 of N , the symbol denotes the rounding off to the next largest integer.

So, for example if the code book contains 512 code words, we need 9 bits to specify which of the 512 code words has been selected at the encoder. The number of bits per input vector of k dimension is therefore, $\log_2 N$ bits.

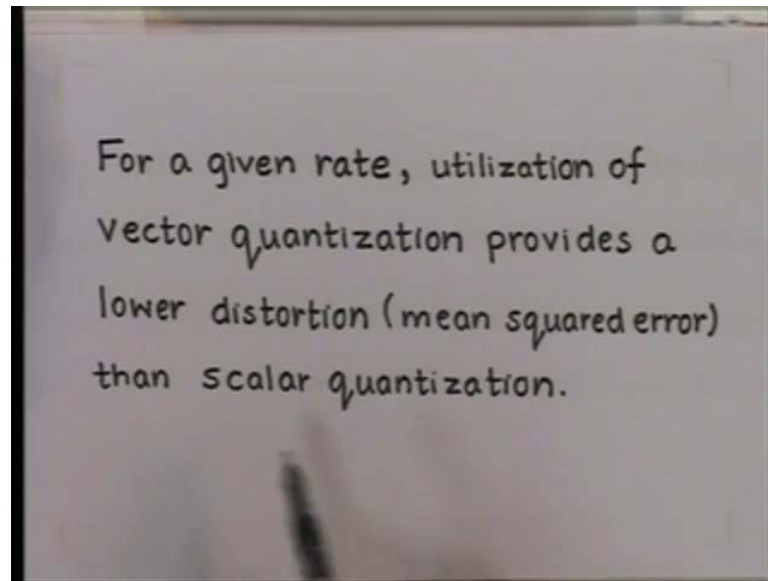
(Refer Slide Time: 10:25)

$$\frac{\lceil \log_2 N \rceil}{k}$$
 Closeness \rightarrow Euclidean distance
 $C \rightarrow \mathcal{N} \{c_j\} \quad X \rightarrow c_i$
 $\|X - c_i\|^2 \leq \|X - c_j\|^2 \quad \forall c_j \in C$
 $X = (x_1, x_2, \dots, x_k)$
 $\|X\|^2 = \sum_{m=1}^k x_m^2$

Now, since each code word contains the reconstruction values for k source output samples, the number of bits per sample would be $\log_2 N$ by k, therefore the rate for an n k dimensional vector quantizer with a code book of size n is given by this expression. Now, as stated earlier too the measure of closeness, which we will use is the Euclidean distance. So, we compute the Euclidean distance between the input vector and the code vector.

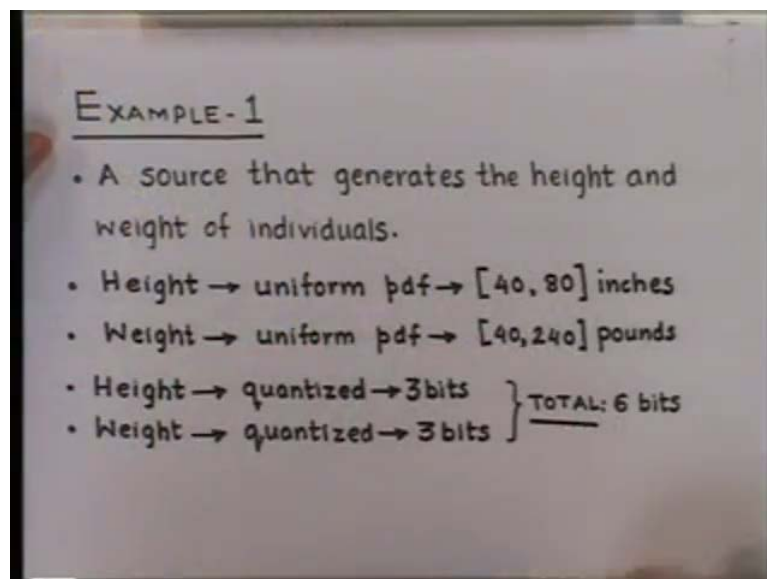
So, when we say that a code book C containing the N code words c_j , the input vector x is closest to code word c_i we mean that Euclidean distance between x and c_i is less than equal to the Euclidean distance between x and c_j , for all c_i belonging to the code book c, where x is the input vector of dimension k. And norm is defined as summation x_n square where n equal to 1 to k.

(Refer Slide Time: 12:53)



Now it is possible to show theoretically, that for a given rate utilization of vector quantization provides a lower distortion, that is mean squared error than scalar quantization. Now, without going into the mathematical proof of the statement, let us get a feel of the advantages of vector quantization over scalar quantization with the help of few examples.

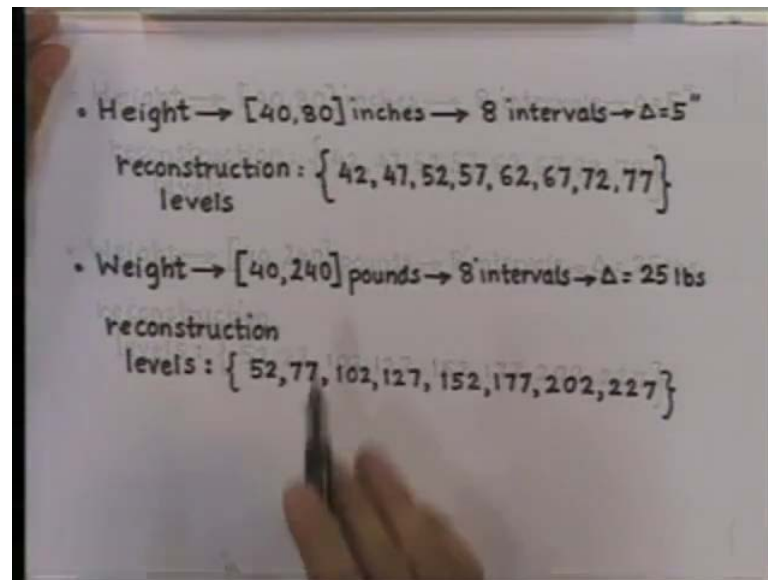
(Refer Slide Time: 13:35)



So, let us consider example one, where a source generate the height and weight of individuals, let us also assume that height is uniformly distributed between 40 and 80

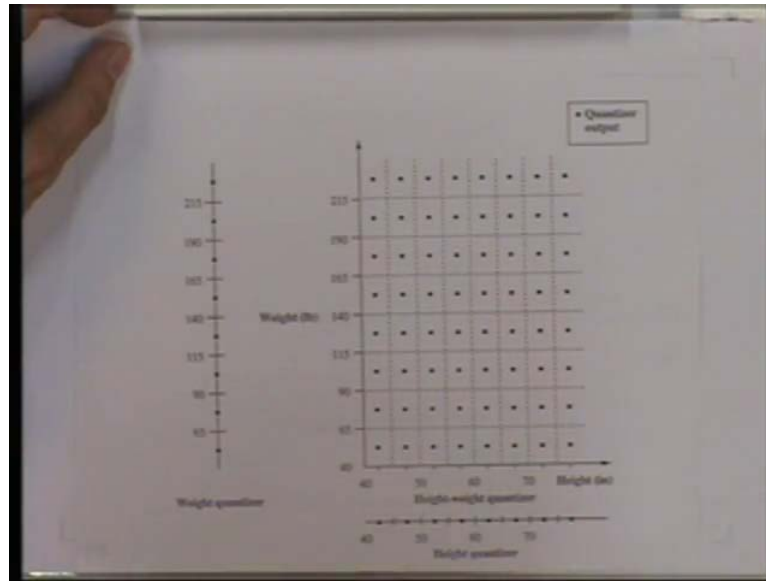
inches. And weight has uniform P D F between 40 and 240 pounds. Now, we have been told that we have total 6 bits to allocate for the quantization of both height and weight of an individual. So, let us allocate 3 bits to quantization for height and 3 bits to quantization for weight.

(Refer Slide Time: 14:38)



So, in this case height with the range between 40 to 80 inches and 3 bits of quantization will give us 8 intervals with the step size of 5 inches. So, the reconstruction levels are indicated here similarly, for weight with the range between 40 to 240 pounds and 3 bits of quantization will give us step size of 25 pounds and the reconstruction levels as shown here. Now, this is what we get when we quantize height and weight individually, this are the individual reconstruction levels. Now, let us examine the same problem in two dimensions. Now, this can be visualized by plotting the height values along the x axis and the weight values along the y axis, and the two dimensional representations of this two quantizers is shown here.

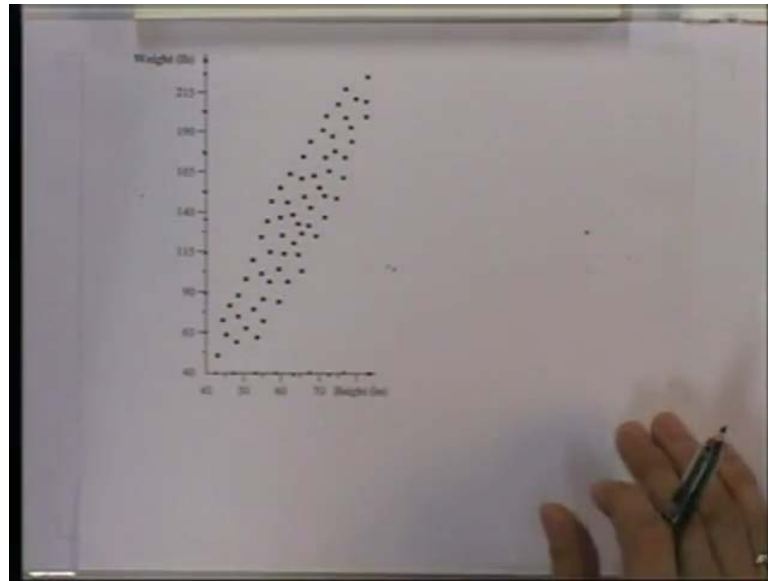
(Refer Slide Time: 15:52)



It is very critical to realize that in this representation, we are not changing anything in the earlier quantization process. The height values are still being quantized to the same 8 different values and so is the case with weight values. What this figure indicates is that we effectively have a quantizer output for a person, who is 78 inches that is 6 feet and 6 inches tall and weighs 40 pounds.

As well as a quantization output for a person whose height is 44 inches, but weighs more than 200 pounds. Now, in a practical scenario with a very high probability obviously, this outputs will never be used and this would be the case for many of the other outputs in this representation. So, what would be a reasonable approach in which to use this quantizer, the answer to this is provided by a solution shown here.

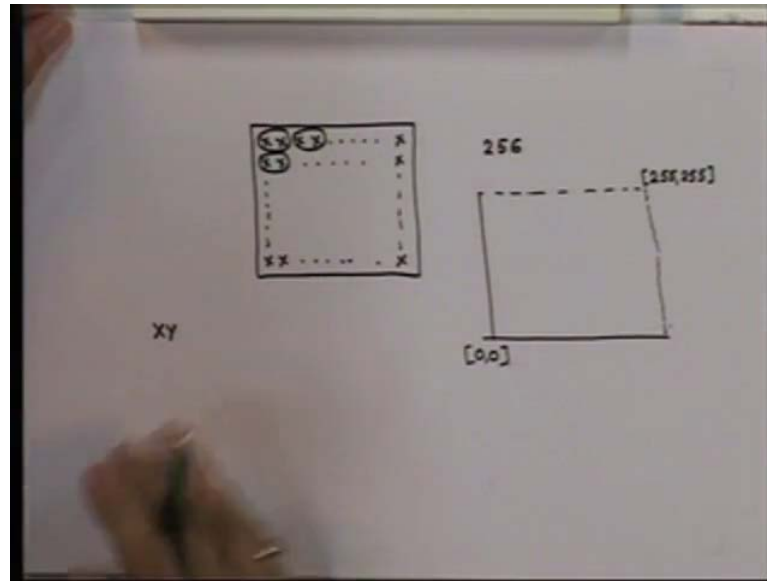
(Refer Slide Time: 17:32)



This quantizer has exactly the same number of output points as the quantizer shown earlier, the difference between this quantizer and this is that in this case, the output points are clustered in the region occupied by the input. Now, using this quantization strategy it is no more possible to quantize the height and weight individually, height and weight are considered as the coordinate of a point in 2 dimensional space, in order to obtain the closest quantizer output point.

All this points are code vectors of code words of a code book of size N equal to 64 because there are 64 points out here. Now, how this code book itself is obtained will be discussed shortly. Now, let us take another example from image compression application, it turns out that vector quantization is a powerful method for lossy compression of data such as sounds, or images because a vector representation often occupies only small fraction of their vector spaces, we can illustrate this distribution in the case of a simple representation of a grey scale image in a 2 D vector space.

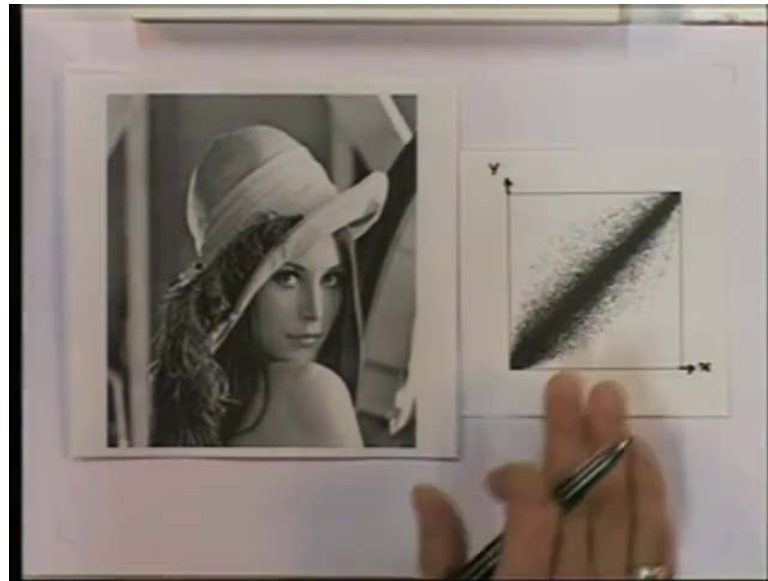
(Refer Slide Time: 19:32)



If we have an image of a particular size and we have the pixels in this image located as follows, then we could compose vectors by taking in pairs the values of adjacent pixels. So, if we scan from left to right top to bottom and take pairs, we can form vectors. Now, if we assume that the input image has 256 shades of grey then we can visualize the vector space as the 0 0 255 255 square in the plane. Now, strictly speaking quantization is the process of approximating continuous values with discrete values.

However, in practice the input values to the quantization process are often also discrete, but with much finer resolution than that of the output values. And the example of this is a grey scale image under discussion. So, we can take two components of the vector as x y coordinates and plot a dot for each vector found in the input image.

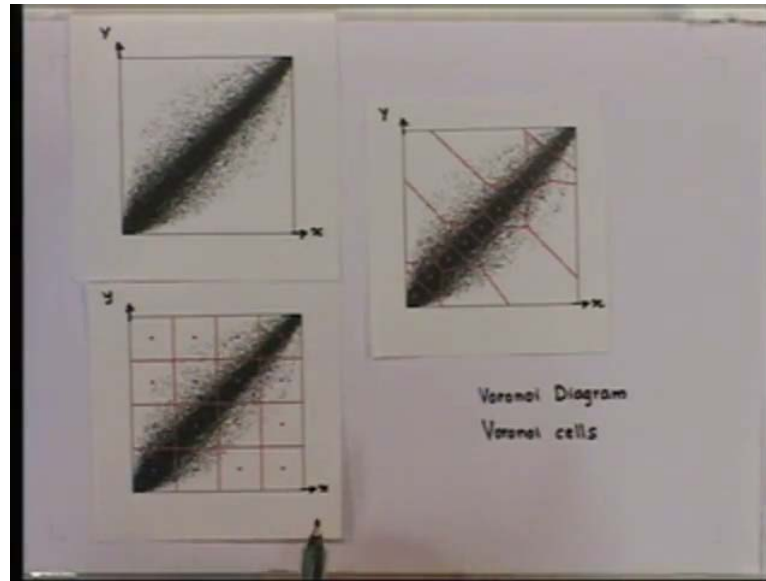
(Refer Slide Time: 21:54)



Now, the figure on my right shows the result of this procedure applied to a grey scale version of the famous Lena image, this is a traditional benchmark for image compression algorithms. So, if we consider this image and form vectors by taking in pairs the values of the adjacent pixels, then this is the plot which we will get. The diagonal line along which the density of the input vector is concentrated is the y equal to x line, the reason for this clustering is that Lena like most photographic images consist predominantly of smooth gradients.

So, adjacent pixels from a smooth gradient will have similar values, and the corresponding dot on the diagram is close to the y equal to x line. The areas on the diagram which would represent abrupt, intensity changes from one pixel to the next are sparsely populated. Now, if we decide to reduce this image to 2 bits per pixel via a scalar quantization, this would mean reducing the pixels to four possible values. Now, if we interpret this as vector quantization on the 2 dimensional vector distribution diagram, then we get a picture shown here.

(Refer Slide Time: 24:13)



The big red dots on the figure represent the 16 evenly spaced possible values of pairs of pixels. Every pair from the input image would be mapped to one of these dots during the quantization. The red lines delimit the zones of influence of they form cells of the vectors, all vectors inside a cell would get quantized to the same code word vector. Now, it is easy to see why this quantization is very inefficient.

Two of the cells are completely empty and four other cells are very sparsely populated, the code book vectors in the 6 adjacent cells to the $y = x$ diagonal are shifted away from the density maxima in their cells, which means that the average quantization error in this cells will be unnecessarily high. Therefore, in other words 6 of the possible of the 16 possible pairs of pixel values are wasted, 6 more are not used efficiently and only 4 are ok.

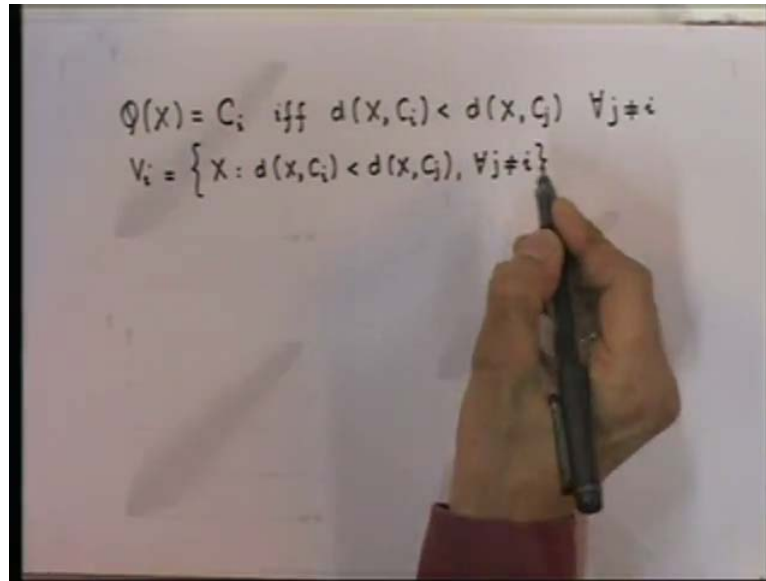
Now, let us perform an equivalent vector quantization for this example. So, instead of 2 bits per pixel we will allocate 4 bits per 2 dimensional vector, but now we can take the freedom to place the 16 vectors indicated by the red dots of the code book anywhere, in the diagram. Now, in order to minimize the mean quantization error we will place all this vectors of code vectors inside the dense cloud around the $y = x$ diagonal. So, if you do that figure here shows how things look vector quantization. As in the earlier figure the code book vectors are represented as big red dots, and the red lines delimit the zones of influence.

This partitioning of a vector space into cells around a predetermined set of spatial vectors, which are called code vectors or code words, such that for all vectors inside a cell, the same spatial vector is closest to them. And this is called a Voronoi diagram and the cells are called Voronoi cells, you can see that in the case of vector quantization the cells are smaller where it matters the most that is the quantization, introduces small errors. And it is in the areas of the vector space where the input vectors are dense. No code book vectors are wasted on unpopulated regions, and inside each cell the code book vector is optimally spaced with regard to the local input vector density.

Now, when you go to higher dimensions for example, taking 4 pixels of vectors instead of pairs better quantization gets more, and more efficient up to a certain point. Now, the question is how to determine the optimal vector size for a given set of input data. Now, this is a rather complicated question beyond the scope of this class, but basically to answer it you need to study the auto look correlation properties of the data. It suffices to say that for the images of the time encountered in practice, 4 is good choice for the vector size. For other applications such as voice compression vector of sizes of 40 to 50 are used.

Now, both this examples demonstrate that in vector quantization, the statistical structure of the source output is exploited by locating the code words in the dense regions of the source output. That is where the source outputs are likely to congregate, from this figure it is also noted that the quantization regions have changed. The decision boundaries between the reconstruction levels can no longer be described as easily as in the case for the scalar quantization. However, if we know the distortion measure simply knowing the output points gives us sufficient information to implement the quantization process. So, in this case instead of defining the quantization rule, in terms of the decision boundaries we can define the quantization rules as follows.

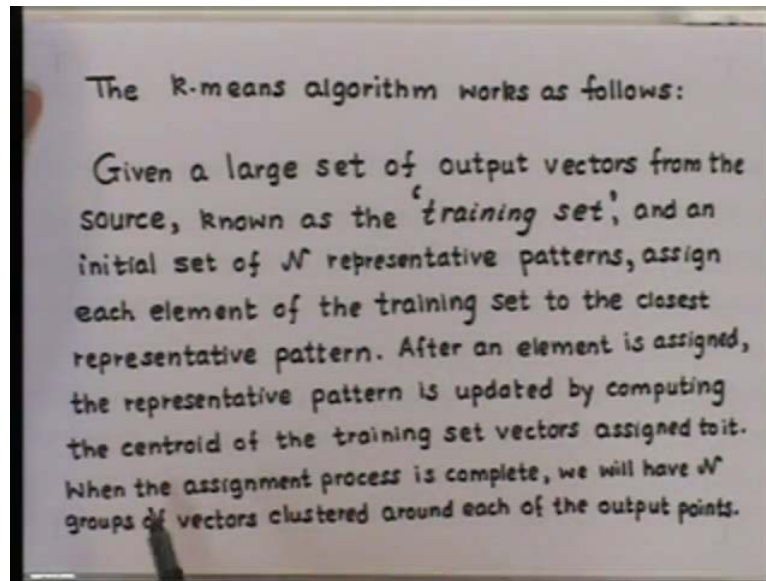
(Refer Slide Time: 31:44)



Q of x is equal to c_i if and only if distance between x , and c_i is less than distance between x and c_j for all j not equal to i . And the quantization regions that is what noise cells denote by V_i can be defined as consisting of all those source output vectors, such that distance between x and c_i is less than the distance between x and c_j , for all j not equal to i , when we group the source output in 2 D case as in the example discussed, we might be able to obtain a good code book design by plotting a representative set of source output points, and then visually locate where the quantizer output point should be. However, this approach to code book design breaks down when we design higher dimensional letter quantizers.

For example, design the code book for a 32 dimensional quantizer obviously, a visual placement will not work in such a case and we need in a algorithm for locating, where the source outputs are clustered. Now, this is a familiar problem in the field of pattern recognition and therefore, the most popular approach to designing vector quantizer is a clustering procedure known as the k men's algorithm, which was developed for pattern recognition applications.

(Refer Slide Time: 34:16)

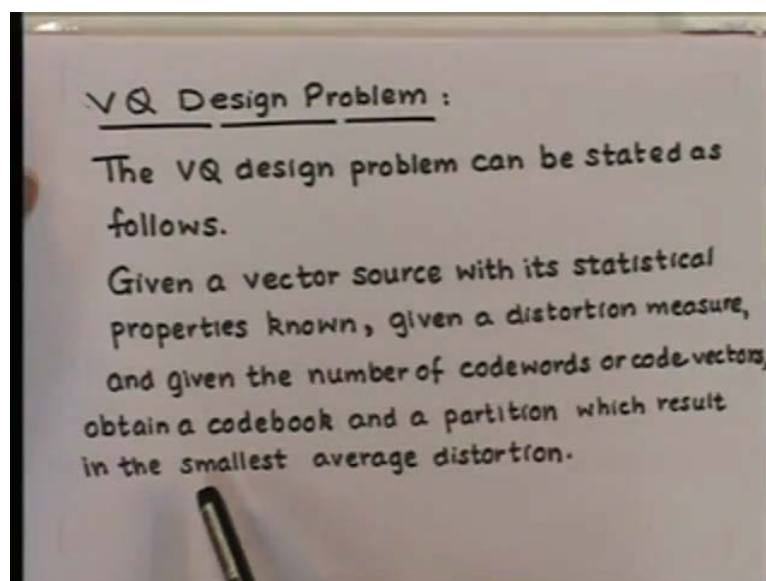


The K-means algorithm works as follows:

Given a large set of output vectors from the source, known as the 'training set', and an initial set of N representative patterns, assign each element of the training set to the closest representative pattern. After an element is assigned, the representative pattern is updated by computing the centroid of the training set vectors assigned to it. When the assignment process is complete, we will have N groups of vectors clustered around each of the output points.

The k men algorithm works as follows. Given a large set of output vectors from the source known as the training set, and an initial set of N representative pattern, assign each element of the training set to the closest representative pattern, after a element is assigned the representative pattern is updated by computing the centroid of the training set vectors assigned to it. When this assignment process is complete, we will have N group of vectors clustered around each of the output points. Now, let us look into a formal mathematical formulation of the vector quantization problem.

(Refer Slide Time: 35:07)



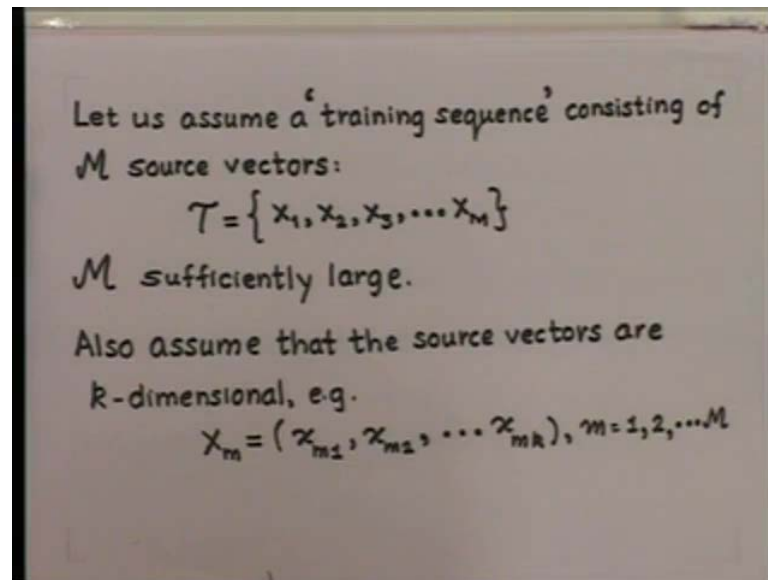
VQ Design Problem :

The VQ design problem can be stated as follows.

Given a vector source with its statistical properties known, given a distortion measure, and given the number of codewords or code vectors, obtain a codebook and a partition which result in the smallest average distortion.

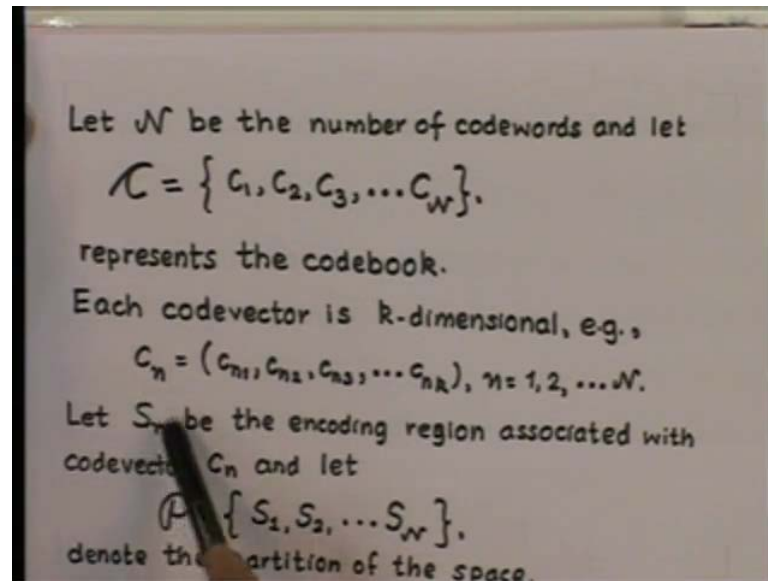
Vector quantization design problem can be stated as follows. Given a vector source with its statistical properties known, given a distortion measure, and given the number of code words of code vectors obtain a code book and a partition which result in the smallest average distortion.

(Refer Slide Time: 35:33)



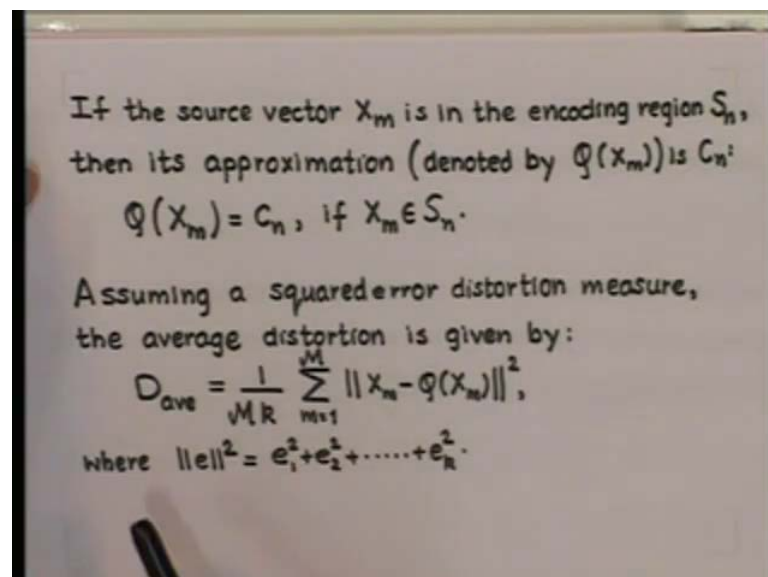
So, let us assume a training sequence consisting of M source vectors as follows. Now, this training sequence can be obtained from some large database. For example, if the source is a p signal, then the training sequence can be obtained by recording several long telephone conversations. For example, if the source is the p signal then the training sequence can be obtained by recording server long telephone conversations. Now, let us assume that M is a sufficiently large so that all the statistical properties of the source are captured by the training sequence. Also always assume that the source vectors are k dimensional. So, each of the source vector is k dimensional.

(Refer Slide Time: 36:48)



Let N be the number of code words and let C denote the set consisting of different code vectors of code word. So, C represents the code book. Now, each code vector is k dimensional given as shown here, and let S_n be the encoding region associated with code vector C_n . And let P be the set consisting of this encoding regions, it denotes the partition of the space and in fact S_n is the Voronoi cell.

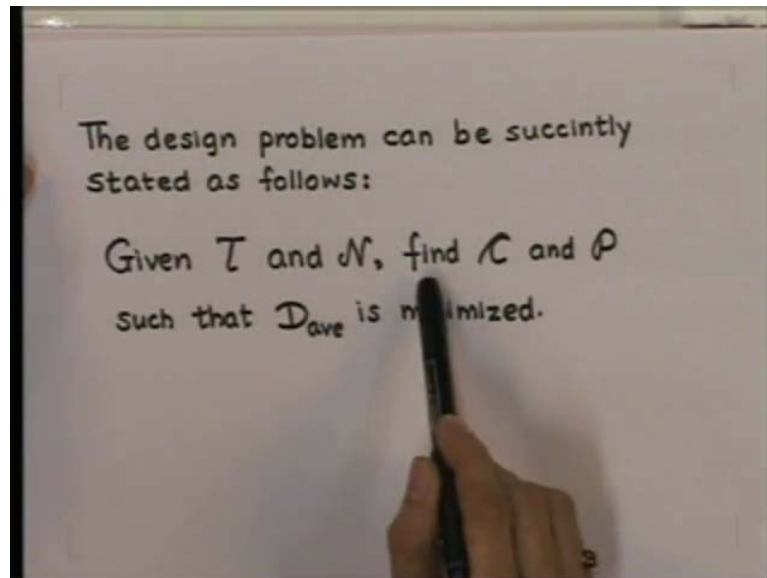
(Refer Slide Time: 37:39)



Now, if the source vector X_m is the encoding region S_n the, it is approximation denoted by $Q X_m$ is C_n mathematically means this. Now, assuming a squared error distortion

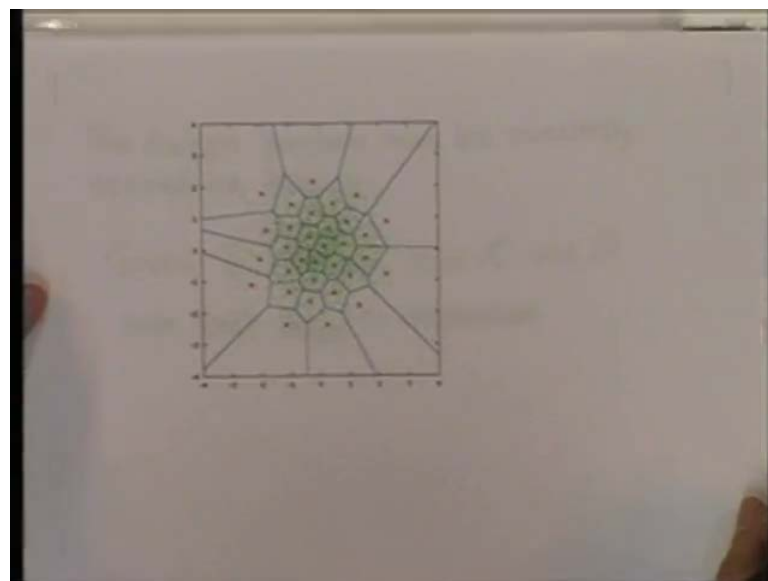
measure the average distortion is given by this expression where the norm has been defined is the Euclidian norm. So, given this the design problem can be succinctly, stated as follows, given a training sequence and the size of the code book.

(Refer Slide Time: 38:19)



That is N find the code book that is set C , and the partition P such that average distortion is minimized as an example.

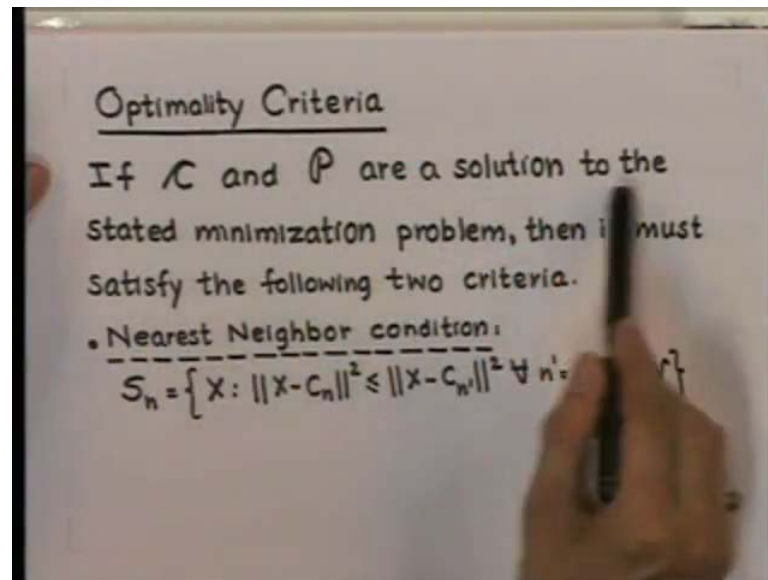
(Refer Slide Time: 38:38)



For 2 dimension case given the green dots in the form of training sequence, and the size of the code book that is N , we need to find the code book that is the code vectors of the

code words denoted by the red asterisks, and the partition P where the boundaries of the Vornoi's cell of this partition are denoted by the blue lines. Now, this code book and partition P , which we obtain as a solution to this problem had to satisfy certain optimality criteria.

(Refer Slide Time: 39:34)



So, if C and P are a solution to the stated minimization problem then it must satisfy, the following 2 criteria. The first criteria is of nearest neighbor condition given by this explanation, what this condition states is that the encoding region S_n should consist of all vectors that are closer to C_n , than any of the code vectors. For those vectors lying on the boundary any of the tie breaking procedure will do, the second condition that should be satisfied.

(Refer Slide Time: 40:14)

• Centroid condition:

$$C_n = \frac{\sum_{x_m \in S_n} x_m}{\sum_{x_m \in S_n} 1}, \quad n=1, 2, \dots, N$$

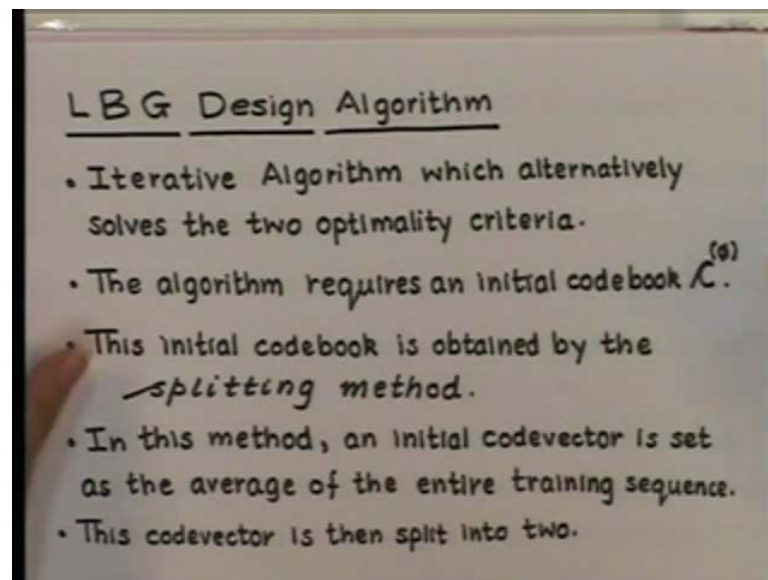
Is of centroid condition given by this expression, this condition states that the code vector C_n , should be average of all those training vectors, that are in the encoding region S_n . Now, in implementation what one should ensure that at least one training vector belongs to each encoding region so that the denominator in this equation is never 0. Now, having discussed the mathematical formulation of vector quantization design problem, let us look at algorithm which provides the solution to this problem. And the most popular is the one given by Linde, Buzo and Gray.

(Refer Slide Time: 41:18)

Y. Linde, A. Buzo, and R. M. Gray,
"An Algorithm for Vector Quantizer
Design",
IEEE Trans. on Communications,
pp. 702-710, January 1980.

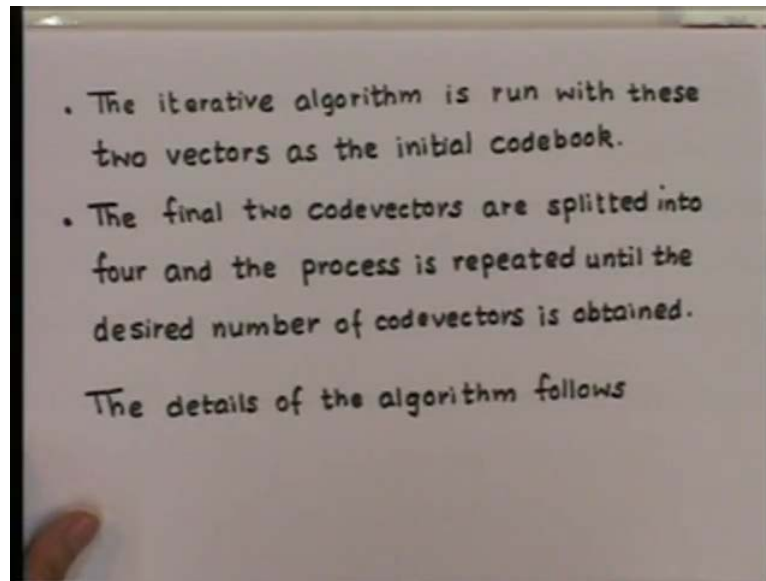
This algorithm appeared in the IEEE transaction on communication in January 1980. Linde, Buzo and Gray have first discussed a generic algorithm, which requires evaluation of multidimensional integrals to compute the distortions and centroid over odd shaped regions in k dimension, where k is dimension of the input vector. Now, generally these integrals are extremely difficult to evaluate and therefore, this algorithm is only of academic interest. Now, Linde, Buzo and Gray in the same paper have discussed another algorithm, which is of practical interest for the case where a training set is available and this algorithm very much resembles the k means algorithm. So, let us discuss this popular algorithm, which is known as LBG design algorithm.

(Refer Slide Time: 42:28)



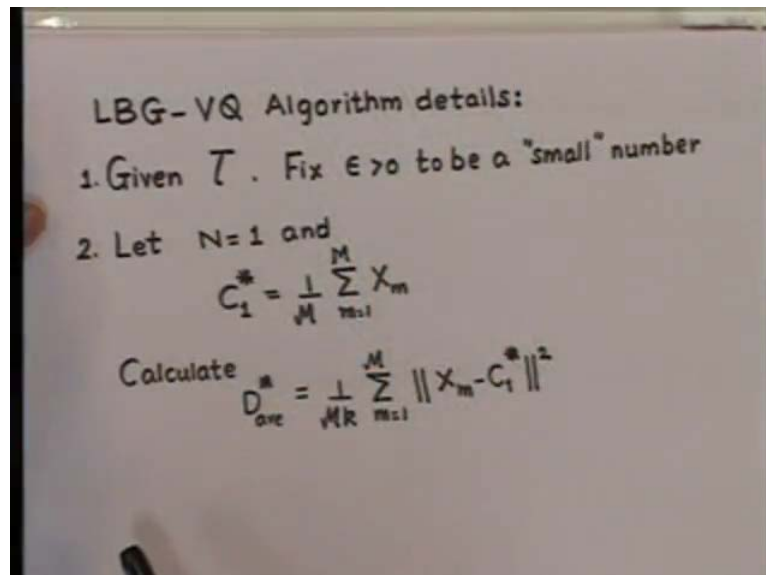
This algorithm is an iterative algorithm, which alternatively solves the two stated optimality criteria. This algorithm requires an initial code book, and this initial code book is obtained by the splitting method. In this splitting method an initial code vector is set as the average of the entire training sequence, and then this code vector is split into two.

(Refer Slide Time: 43:03)



Now, the iterative algorithm is run with this two vectors as the initial code book and then the final two code vectors are splitted into four, and the process is repeated until the desired number of code vectors is obtained. Now, the details of the algorithms follow we have been given the training sequence.

(Refer Slide Time: 43:24)

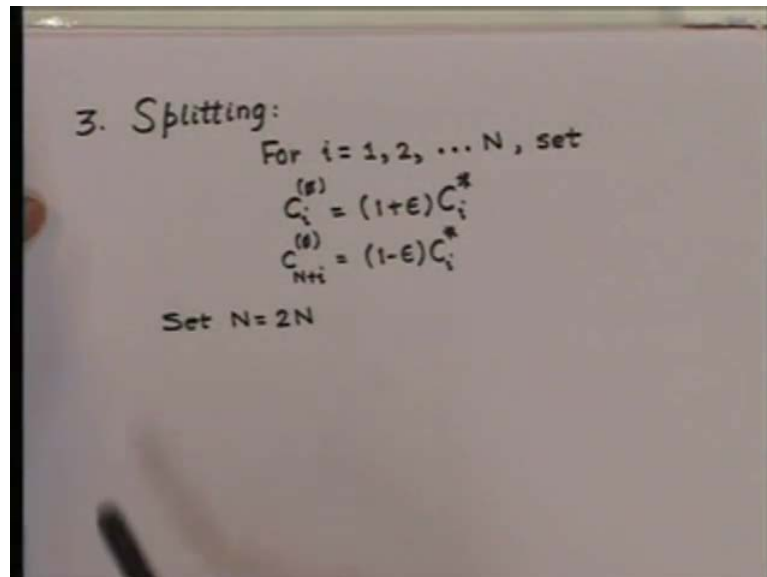


We fix an epsilon which is greater than 0 to be a small number to start with let us assume that the code book size is 1. So, we set N equal to 1 and for this code book the only

optimal code vector which is denoted by C_1^* is obtained by the average of the complete training set.

Now, if the code book size was of one we would have stopped at this point, but this would be a very gross approximation of the complete training set. And in this case the average distortion would be given by this expression. So, let us assume that the code book size required is larger than 1. So, the next step is to obtain a code book and partition for code book size 2, and in order to do this we require two initial code vectors, and this is obtained by the splitting procedure.

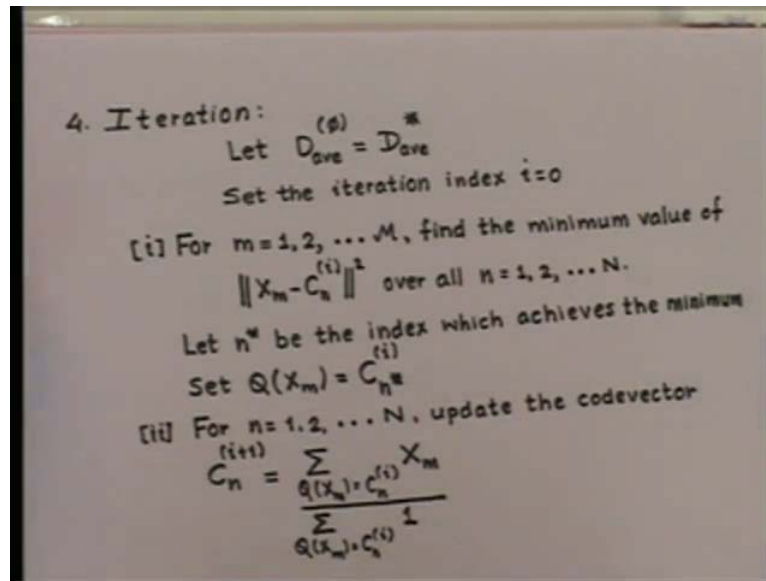
(Refer Slide Time: 44:38)



And the splitting procedure is indicated here now to start with our N is equal to 1 therefore, i is equal to 1 and we split the earlier code word that is C_1^* into two initial code vectors given by C_{10} equal to $(1+\epsilon)C_1^*$ and C_{20} equal to $(1-\epsilon)C_1^*$. For our discussion we will read these quantities as $C_i^{(s)}$ or $C_{N+i}^{(s)}$.

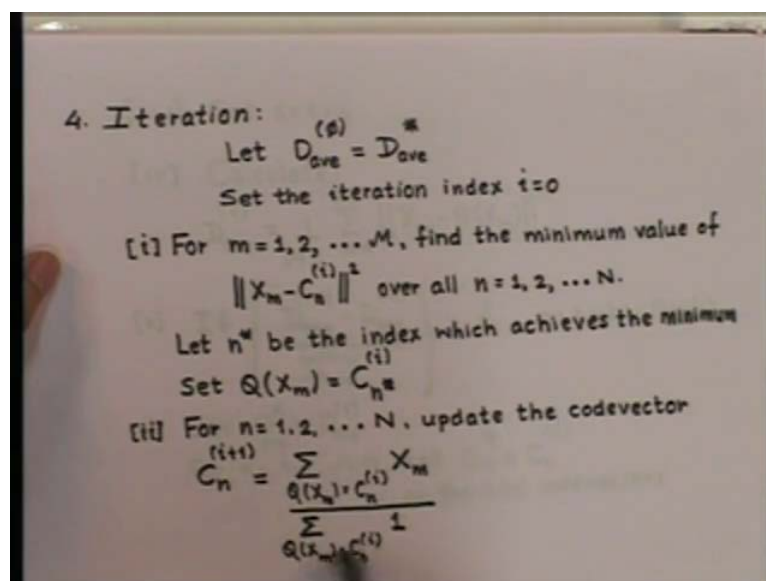
Now, we have two initial code vectors in this code book therefore, a size of the code book has increased to 2. So, N is equal to 2. Now, if the next step is to obtain the optimum code book and partition for the code book size 2, and this is done by moving it over to the next step of iteration. So, we first set the average distortion D_0 that is initial average distortion as the average distortion obtained in the last step.

(Refer Slide Time: 45:54)



And set the iteration index i equal to 0. Now, there are 6 sub steps in this step and the first sub step is given here. Now, this sub step divides the complete training set into 2 groups using the following strategy. So, in principle it assigns each source output vector in the training set to one of the initial code vector, and at the end of this sub step all the source output vectors, in the training set are assigned to one of the two initial code vectors. The next sub step would be to obtain the updated code vectors that is C_{11} and C_{21} given by this expression.

(Refer Slide Time: 47:14)



Next, we set the iteration index to i equal to 1 and in this step we calculate the average distortion given by $D_{ave}^{(i)}$ using this expression, where we use the updated code vectors C_{11} and C_{21} . Now, it is expected that this average distortion will be less than the average distortion, which we got in the earlier iteration and so we calculate the difference between the current distortion and the earlier distortion.

And from this ratio and compare this ratio to some small δ greater than 0. If this quantity is greater than δ it implies that we need some more refinements. So, we go to step one where we again divide the complete training set into two groups, but this time using the updated code vectors C_{11} and C_{21} at the end of this sub step, we again get the updated code vector C_{12} C_{22} .

(Refer Slide Time: 48:46)

Handwritten notes on a whiteboard showing the iterative process of calculating average distortion and updating code vectors:

- [iii] Set $i = i + 1$
- [iv] Calculate:

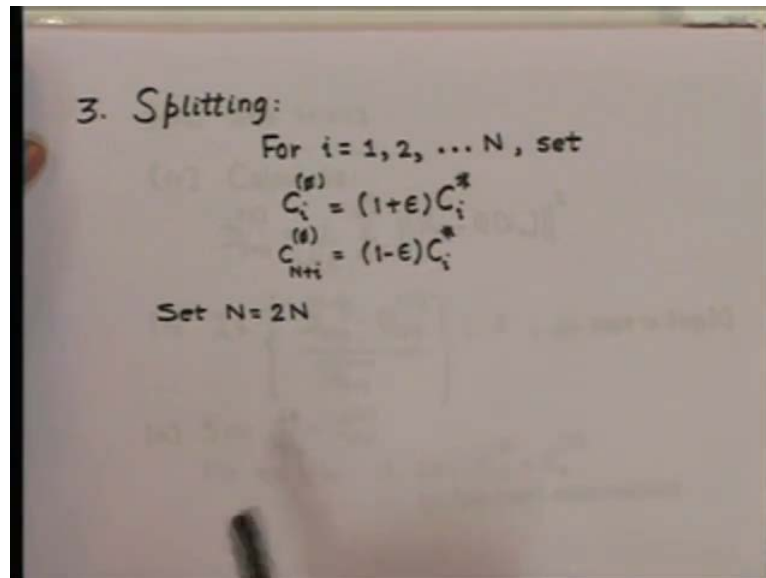
$$D_{ave}^{(i)} = \frac{1}{M} \sum_{m=1}^M \|x_m - Q(x_m)\|^2$$
- [v] If $\left\{ \frac{D_{ave}^{(i-1)} - D_{ave}^{(i)}}{D_{ave}^{(i-1)}} \right\} > \delta$, go back to Step [i]
- [vi] Set $D_{ave}^* = D_{ave}^{(i)}$
For $n = 1, 2, \dots, N$, set $C_n^* = C_n^{(i)}$ as the final codevectors

And then average distortion at the step number 2 will be given by this expression, where we use the updated code vectors C_{12} and C_{22} , again calculate this ratio. Now, if this ratio is less than δ we come out of the loop, which implies that we have obtained the optimum code book and partition for the code book size of 2. At this stage we set the final average distortion indicated by D_{ave}^* equal to average distortion at the i th iteration. That is the one which we had calculated just before exiting that is the one which we had calculated just before exiting the loop at step number 5.

And similarly, we set the final code words C_n^* equal to the calculated or updated code vectors, which existed before exiting out of the loop at step number 5. Now, if the

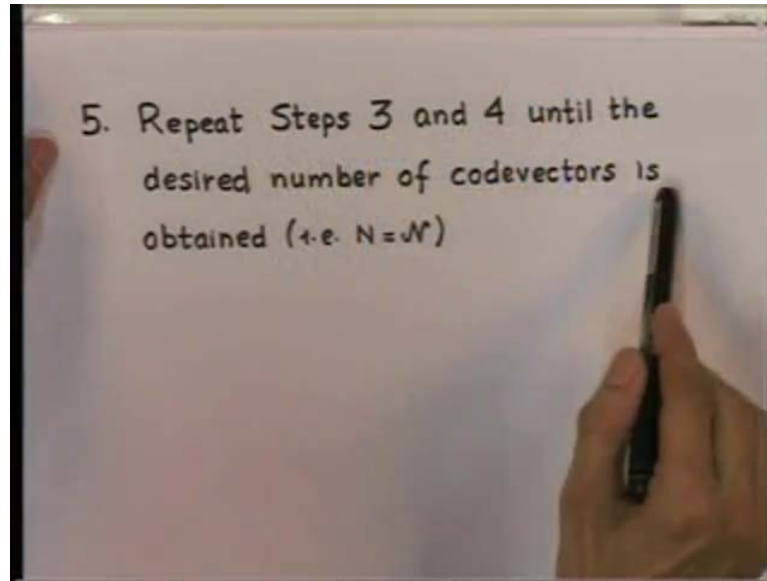
code book size was 2 we would stop at this, but if the code book size was larger than 2 then we would go to design an optimum code book. And partition for code book size of 4 which requires that we have 4 initial code vectors, and to get those 4 initial code vectors.

(Refer Slide Time: 50:26)



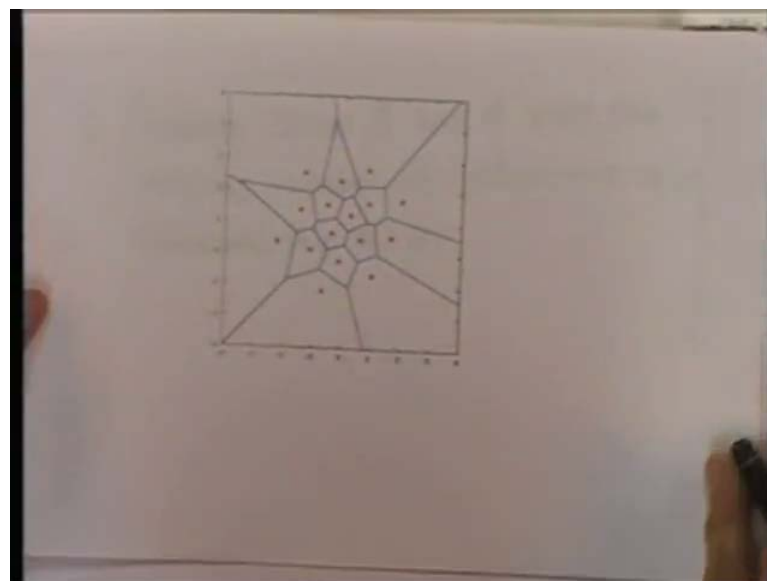
We go this step of splitting, where now your N has become 2. So, for i equal to 1 to 2 we have to compute this. So, we get $C_1^{(s)}$ as $(1+\epsilon)C_1^*$ and $C_3^{(s)}$ equal to $(1-\epsilon)C_1^*$ and $C_2^{(s)}$ equal to $(1+\epsilon)C_2^*$ and $C_4^{(s)}$ equal to $(1-\epsilon)C_2^*$. So, we have now 4 initial code vectors and using these 4 initial code vectors we have to get the optimum code book, and partition for code book size 4 and to get that we again move over to the step number 4, that is iteration and once more we go through this 6 sub steps under the iteration steps. And this way we repeat steps 3 and 4.

(Refer Slide Time: 51:33)



Until the desired number of code vectors is obtained.

(Refer Slide Time: 51:40)

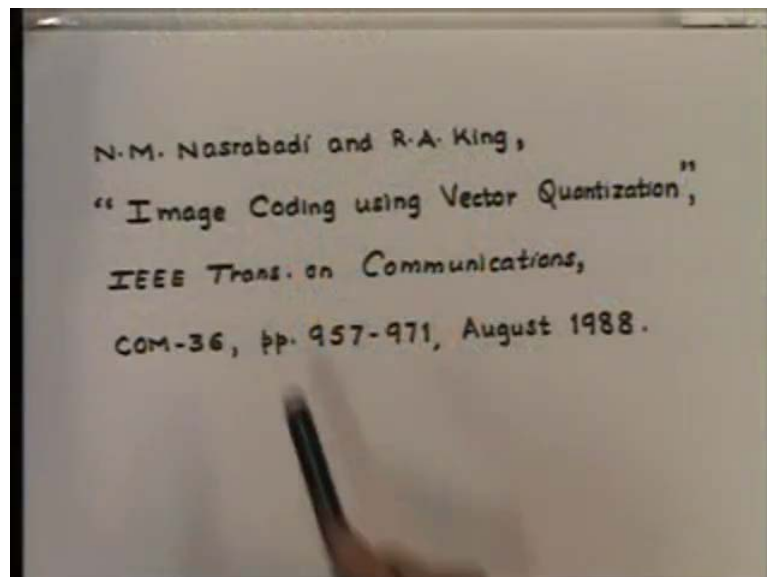


So, as an example for the case of two dimension, finally we will obtain the code book in the form code words or code vectors indicated by the red asterisk, and Voronoi regions whose boundaries are indicated by the blue lines. So, in summary vector quantization is a multidimensional approach to quantization, and from this perspective scalar quantization restricts the output points of the quantizer to rectangular regions, but by observing source output samples in terms of vectors of some length k , it provides the flexibility to locate

or to place the output points of the quantizer in appropriate positions in the k dimensional space.

Now, the three important issues which are related to the performance of the vector quantizer are code book design, the algorithm used by the search engine to find the closest match between the input vector, and the code vector in the code book. And finally, the statistical characteristic of the input vector itself. Now, in the literature there are many variants of vector quantizer, which have been discussed and which address this issues which I just mentioned, and one such reference is shown here.

(Refer Slide Time: 53:26)



This is a reference by Nasrabadi and King which appeared in i triple E transaction on communications, and this reference discuss some of the basics of different variance of vector quantization. Now, in the next class we will discuss another block quantization technique in which the output source vector is decomposed or transformed into components that are then coded according to the individual characteristics. We will discuss the issues of quantization and coding of these transformed coefficients.