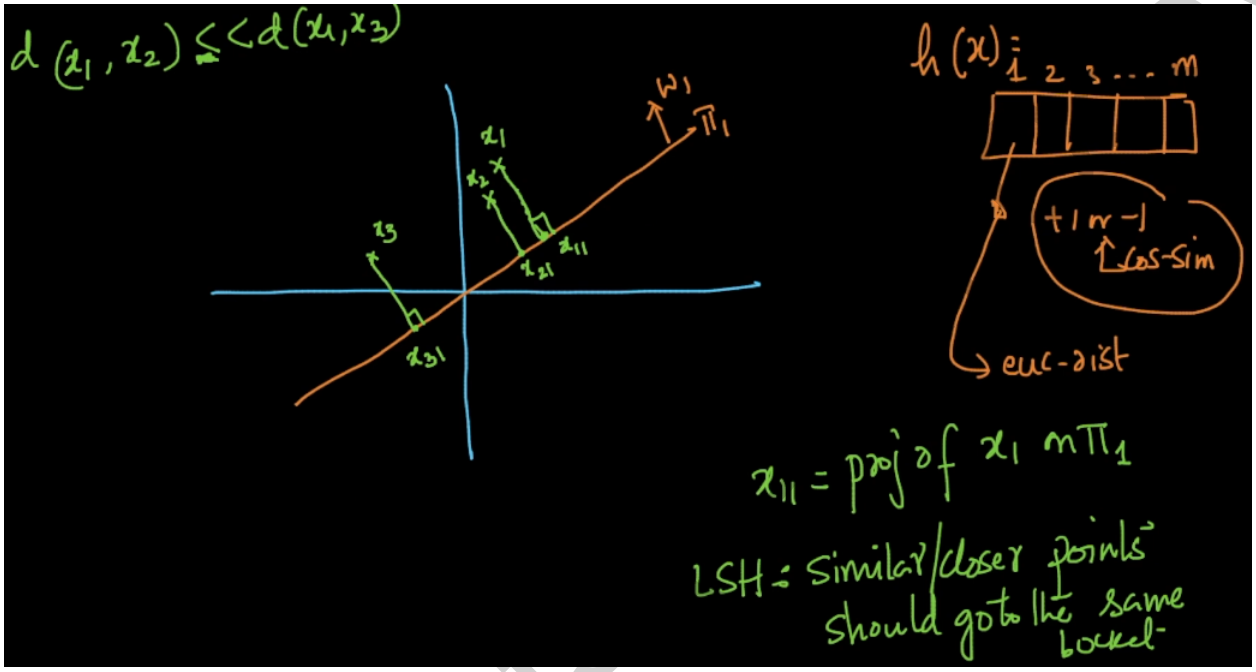
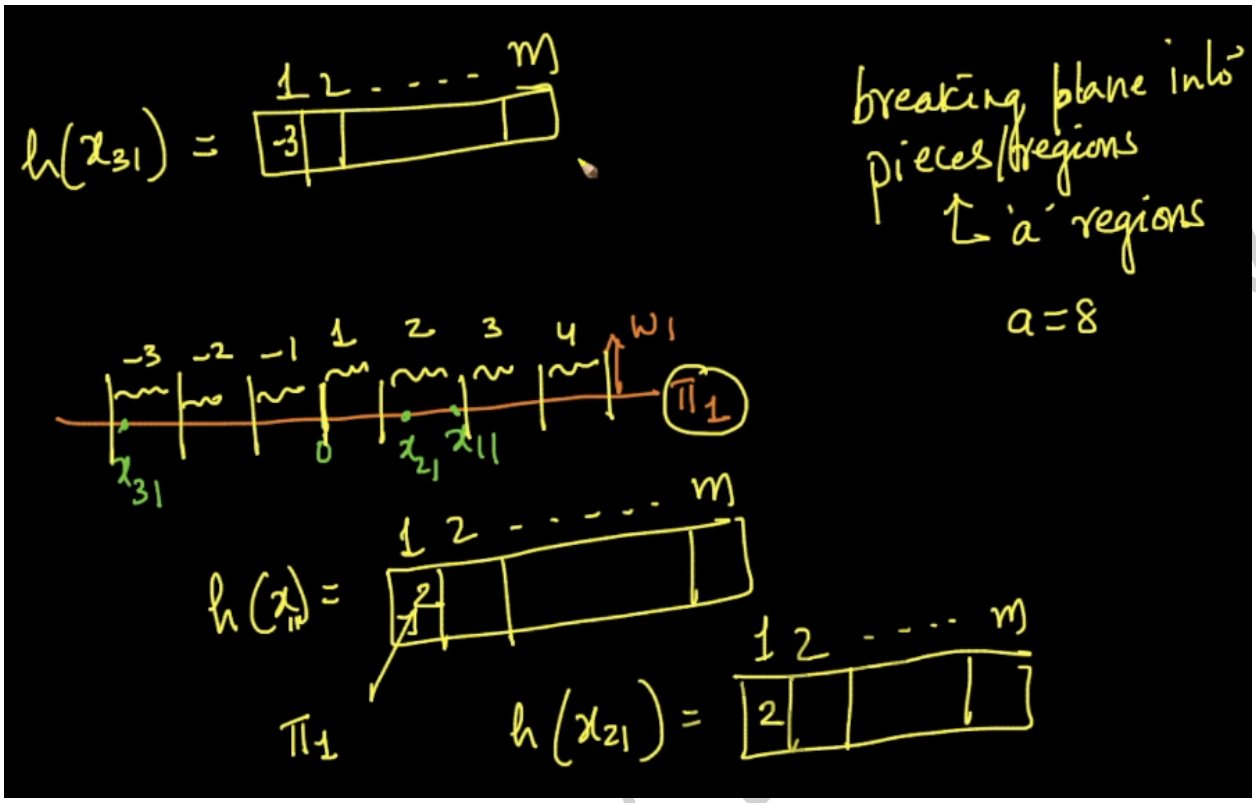
29.28 LSH for Euclidean Distance

So far we have seen that if a point 'x_i' belongs to the region same as 'w1', then we use +1, and if it is present on the opposite side, we use -1 in the hash key.

Now we shall draw the perpendiculars of all the points onto the hyperplane 'π', as shown below.



We now have to divide this hyperplane into the regions on the basis of the projections of the points on it. We shall give labels to each region as shown below.



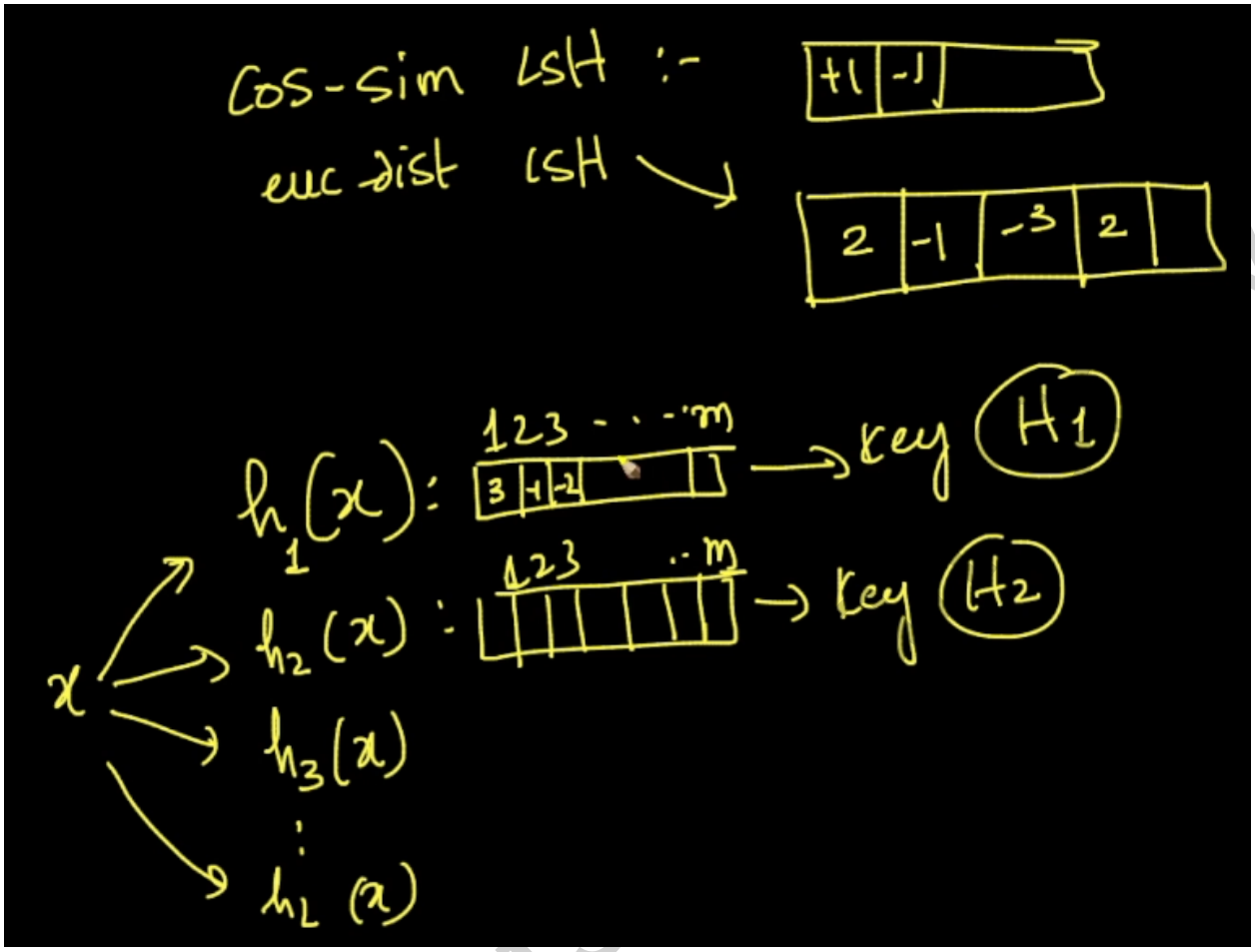
Now we have to create the hash key. If there are 'm' hyperplanes, then the hash key would be a 'm' dimensional vector.

The hash key for the point 'x_{11}' consists of '2' for the component '1'. (It is because this point is having a projection onto the hyperplane 'π_1'. So the 1st component of the 'm' dimensional vector would be filled with the value. This value here would be the region number in which this point is present). We can see the point 'x_{11}' lies in region '2' of the hyperplane 'π_1', so the first component of the hash key would be '2'.

When it comes to the hash key of 'x_{21}', even here this point lies in the 2nd region of the hyperplane 'π_1'. So the first component of the hash key should be '2'.

If we look at the hash key of the point 'x_{31}', as this point is present in the region '-3', we fill the first component of the hash key with '-3'.

This is the difference between LSH for Cosine Similarity and LSH for Cosine Distance. The hash values in LSH for Cosine Similarity contain +1 or -1, whereas in LSH for euclidean distance, the value can be anything(the region number associated with the hyperplane).



Let us assume, here is LSH for euclidean distance, we have 'L' hashtables, then for every point 'x', we have to generate 'L' hash keys, as shown above. Each of these hash keys is 'm' dimensional.

$h_1(x)$ → Hash key of the point 'x' in Hashtable 1

$h_2(x)$ → Hash key of the point 'x' in Hashtable 2

.

.

$h_L(x)$ → Hash key of the point 'x' in Hashtable L

The points that have the same region number tend to be closer. If 2 points have the same region value of a hyperplane 'π_1', then the distance between them will be closer.

But again as these hyperplanes are formed randomly, they can be formed in any direction, and due to this sometimes the points may fall in different regions.

So LSH is not perfect all the time. It is a probabilistic/randomized algorithm and is still used extensively.