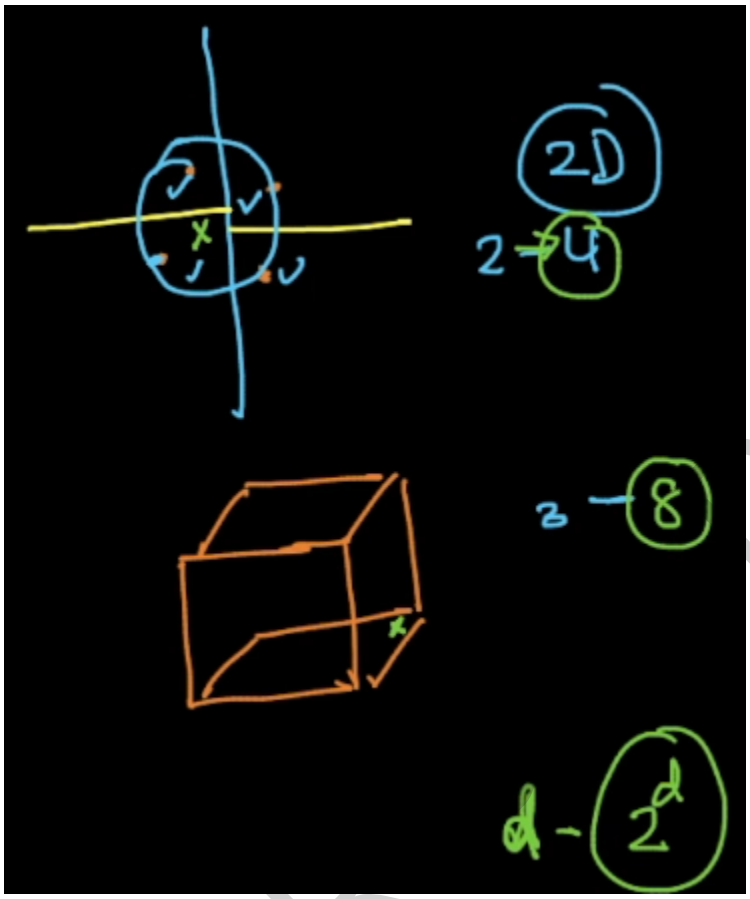
29.24 Limitations of KD-Tree

So far we have seen that if 'd' is small,
Best Case Time Complexity for KNN = O(k*log(n))
Worst Case Time Complexity for KNN = O(k*n)



So far, we have seen when our data consists of 2-dimensions, we have to look up $2^2 = 4$ adjoining cells.

If the data is in 3-dimensional form, then the number of adjoining cells to look up $= 2^3 = 8$.

So if the data is in d-dimensional form, then the number of adjoining cells to look up $= 2^d$
For example, if the number of dimensions = 20, then
Number of adjoining cells to look up = 2^{20}
Due to this, as the 'd' value is not small, the time complexity is no more O(log(n)).
The time complexity changes to $O(2^d*\log(n))$.
If $2^d = n$, then the time complexity is O(n*log(n)).
As the number of dimensions increases, the number of adjoining cells also increases, thereby increasing the time complexity drastically. In such a case, KD-Tree is useless.

The Space Complexity of KD-Tree = O(n)
Even when the dimensionality is small, O(log(n)) time complexity holds good only if our data is uniformly distributed in space. If the data is not uniformly distributed, the time complexity changes to O(n). (same as that of brute force KNN).

Note: When compared to the other models, KNN is often used as a baseline model, when we are operating with small data. KD-Tree is not developed to find the nearest neighbor in Machine Learning, but was developed to find the nearest neighbors in the computer graphics.

29.25 Extensions

Refer to the wikipedia article link given below, as this video lecture is purely theory based, and also the content in it was taught from the below mentioned link only.
Wikipedia Link: https://en.wikipedia.org/wiki/K-d_tree#See_also
For any queries regarding this topic, please feel free to post them in the comments section below the video lecture.