To address the inefficiencies of KD Trees in higher dimensions, the *ball tree* data structure was developed. Where KD trees partition data along Cartesian axes, ball trees partition data in a series of nesting hyper-spheres. This makes tree construction more costly than that of the KD tree, but results in a data structure which can be very efficient on highly structured data, even in very high dimensions.

A ball tree recursively divides the data into nodes defined by a centroid x and radius y, such that each point in the node lies within the hyper-sphere defined by x and y. The number of candidate points for a neighbor search is reduced through use of the *triangle inequality*:

|x+y|≤|x|+|y|

With this setup, a single distance calculation between a test point and the centroid is sufficient to determine a lower and upper bound on the distance to all points within the node. Because of the spherical geometry of the ball tree nodes, it can out-perform a *KD-tree* in high dimensions, though the actual performance is highly dependent on the structure of the training data. In scikit-learn, ball-tree-based neighbors searches are specified using the keyword algorithm = 'ball\_tree', and are computed using the class [**BallTree**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html#sklearn.neighbors.BallTree). Alternatively, the user can work with the [**BallTree**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html#sklearn.neighbors.BallTree) class directly.