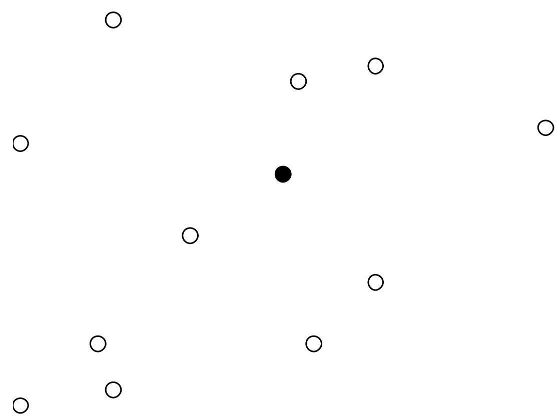
NEAREST NEIGHBOR SEARCHES



The Problem: Given a set of points **C** (the white circles) and a “query” point **X** (the black circle), find which member of **C** is closest to **X**.

The Real Challenge: Write an algorithm to solve this problem in sub-linear time.

APPLICATIONS

Nearest neighbor search problems occur frequently in practice.

They can be used to find similar objects in many different domains. For example,

- Text Strings,
- Web Pages,
- DNA Sequences,
- Images, and
- Audio.

They are also occur as the central component of important algorithms in areas such as

- Data Mining,
- Pattern Recognition, and
- Data Compression.

DATA COMPRESSION

(Compression Ratio approx. 1 : 225)

Vector Quantization: Each block of pixels is considered to be a k-dimensional vector. (You can see the blocks of pixels in the picture above.) Select a small group of representative blocks. Then replace each block in the original image with the most similar representative.

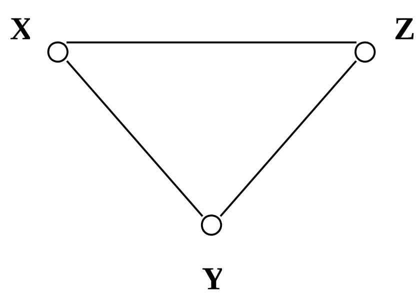
GENERAL PROBLEM

We are given

- a set **C** of vectors (in k-dimensions),
- a query point **X**,
- a guess $\mathbf{C_G}$ of the nearest neighbor,
- and a distance measuring function **d**(**X**,**Y**).

The distance function is sometimes called a distance metric, distortion measure, or a dissimilarity measure.

In general, there are no restrictions we can put on the distance function. We will assume only that **d** obeys the triangle inequality:



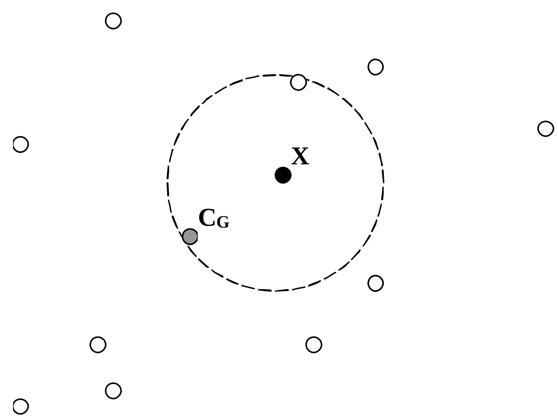
$$\mathbf{d}(\mathbf{X},\mathbf{Z}) \leq \mathbf{d}(\mathbf{X},\mathbf{Y}) + \mathbf{d}(\mathbf{Y},\mathbf{Z})$$

However, consider some of the weird distance functions like the editing distance between two text strings. Does it obey the triangle inequality?

METHOD ONE: K-D TREES

One reasonable method for performing nearest neighbor searches would be to store the points in a k-D tree.

To find the nearest neighbor, we would need to check all of the points inside the circle of radius $\mathbf{d}(\mathbf{C_G},\mathbf{X})$ centered at $\mathbf{X}$. This is just a circular range query.



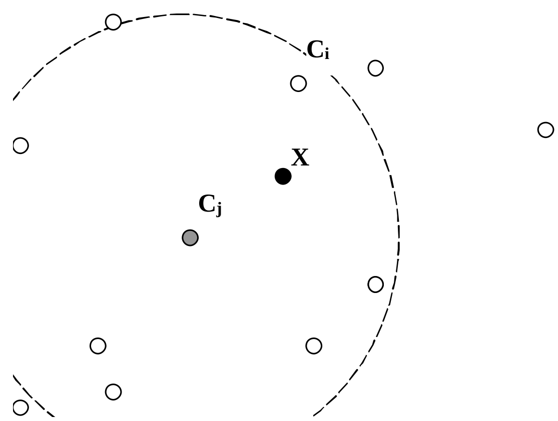
Bentley proved that these range queries are O(log n) on average for any distribution of points.

What is their worst case complexity?

METHOD TWO: ORCHARD'S METHOD

Orchard's Method is based on the following geometric fact (which can be proved by the triangle inequality): if $\mathbf{C_i}$ is closer to $\mathbf{X}$ than $\mathbf{C_j}$, then it must be the case that

$$\mathbf{d}(\mathbf{C_i},\mathbf{C_j}) < 2 \times \mathbf{d}(\mathbf{C_j},\mathbf{X})$$



Since we can pre-compute a list of the items in $\mathbf{C}$ sorted by their distance from $\mathbf{C_j}$, all we have to do at run time is move down the list until we get to an item $\mathbf{C_i}$ for which $d(\mathbf{C_j},\mathbf{C_i}) > 2 \times \mathbf{d}(\mathbf{C_j},\mathbf{X})$.

How do we implement this?

What is the biggest drawback of this technique?

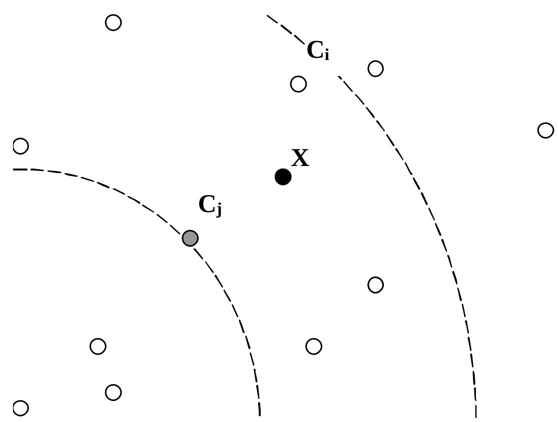
METHOD THREE: ANNULUS METHOD

The Annulus Method is another geometric fact that can be proved from the triangle inequality: if $\mathbf{C_i}$ is closer to $\mathbf{C_j}$, then it must be the case that

$$\mathbf{d}(\mathbf{X},\mathbf{P}) - \mathbf{d}(\mathbf{C_j},\mathbf{X}) \leq \mathbf{d}(\mathbf{C_i},\mathbf{P}) \leq \mathbf{d}(X,\mathbf{P}) + \mathbf{d}(\mathbf{C_j},\mathbf{X})$$

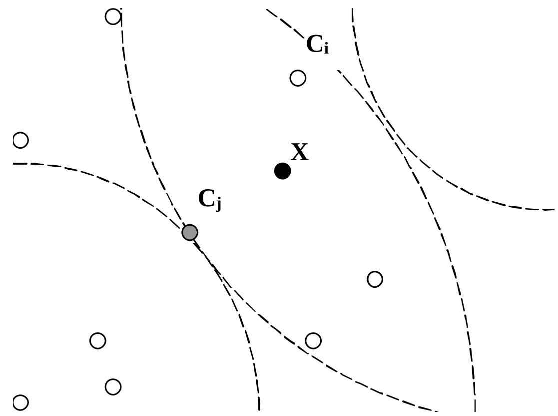
where **P** is some arbitrary point (most often the origin, in which case $\mathbf{d}(\mathbf{C_i},\mathbf{P})$ is the norm of $\mathbf{C_i}$).

This equation gives us an annular region in which to search.



How do we implement this?

We can extend this method by computing the distances of each member of **C** to two other points instead of just one. With that information, we can then form two annular regions.



We need only search those items that lie in both annular regions.

How should we implement this method?

This technique, called the Double Annulus Method, seems to be faster. However, the difference between them appears to only be a constant factor.

GUESSING

Here are at three techniques for picking $\mathbf{C_G}$.

- **Pure Prediction**: Guess that the next nearest neighbor is going to be the same as the last.
- **Norm Prediction**: Pick $\mathbf{C_G}$ to be the member of $\mathbf{C}$ whose norm is closest to the norm of $\mathbf{X}$.
- **Tree Prediction**: Traverse the k-D tree down to the point where $\mathbf{X}$ would be inserted and pick $\mathbf{C_G}$ to be the member of $\mathbf{C}$ that is in that place.