Weighted and Continuous Clustering

John Burkardt (ARC/ICAM)
Virginia Tech

....

Math/CS 4414:
"Weighted and Continuous Clustering"
http://people.sc.fsu.edu/∼jburkardt/presentations/
clustering_weighted.pdf

....

ARC: Advanced Research Computing
ICAM: Interdisciplinary Center for Applied Mathematics

25 September 2009

Weighted and Continuous Clustering

The K-Means algorithm is a simple and very useful tool for making an initial clustering of data.

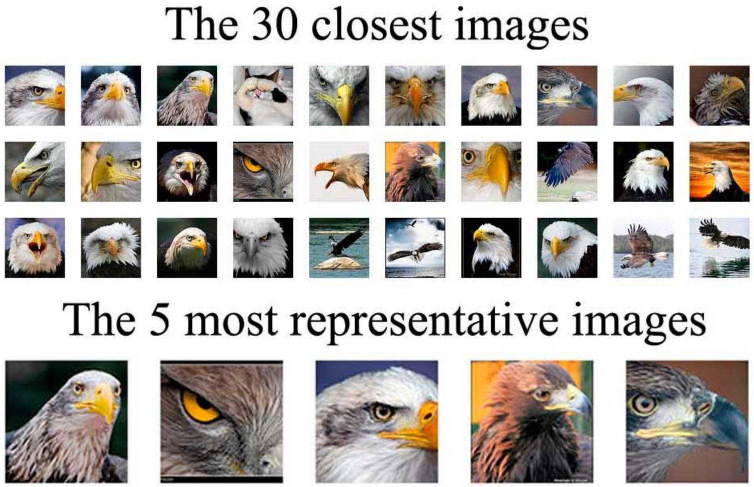
It does have the drawback that often stops at an answer that isn't the best one, but this can be detected and corrected by running the program with several different starting points.

Because it can give some information about almost any set of data, researchers have tried to extend it to handle new classes of problems for which good clusterings are hard to find.

As an example, online image repositories may have thousands of images classified as images of "an eagle". It is important to be able to return a few "typical" images.

An Extension of K-Means

An Extension of K-Means



Weighted and Continuous Clustering

Today we will consider some extensions of the K-Means algorithm that are not so ambitious but which let us consider new classes of problems.

If we know that some data has a greater importance, we would like to be able to include that information, so that the clustering can take that into account.

When you cluster hundreds of points in a region, the points are organized into polygonal subregions. Can we just go ahead and cluster "all" the points in, say, the unit square? Instead of a discrete set, we have a continuous set of points to consider.

Weighted and Continuous Clustering

Weighted K-Means: The Centroid

Suppose that we want to cluster a set of data points **X**, but now we want to be able to stipulate that some points are more "important" than others.

We might encounter this problem if we are deciding where to put the capital city of a new state, or the hubs of an airline, or a brand new set of elementary schools in a city.

In each of these cases, we are trying to choose convenient "centers", but now it's not just the geographic location that we have to consider, but also *how many* people or average flights or school children will have to travel from the data points to the cluster centers.

Weighted K-Means: Using Weights

The natural way to record the varying importance of the different points would be to assign weights to each one. Thus, our original problem can be thought of as the simple case where each point was given the same weight.

With the addition of weights, the K-Means algorithm can handle situations in which some data points are so important that they almost pull one of the cluster centers close to them by gravitational attraction.

Weighted K-Means: The Centroid

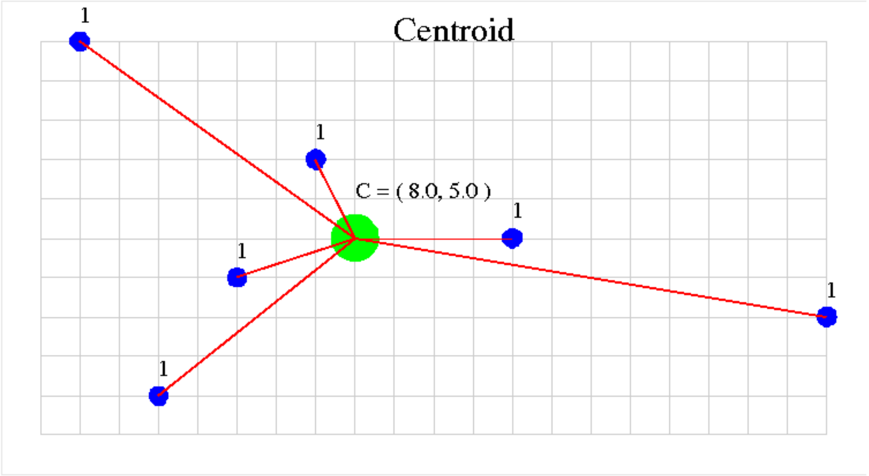
In the unweighted case, a cluster center $\mathbf{c_j}$ was the **centroid** or average of the coordinates of all the data points $\mathbf{x_i}$ in the cluster:

$$\mathrm{c_j} = \frac{\sum_{x_i \in c_j} x_i}{\sum_{x_i \in c_j} 1}$$

The centroid is a geometric quantity whose location can be determined from just a picture of the points.

If we imagine the points being connected to the centroid, and having equal weight, then this object is perfectly balanced around the centroid. No matter how we turn it, it will be balanced.

Weighted K-Means: The Centroid



Weighted K-Means: The Cluster Variance

For unweighted clustering, we defined the *cluster variance* as:

$$\begin{aligned}\mathrm{var}(\mathrm{x},\mathrm{c}) &= \sum_{j=1}^{k} \mathrm{var}(\mathrm{x},\mathrm{c_j}) \\ &= \sum_{j=1}^{k} \frac{\sum_{x_i \in c_j} \|x_i - c_j\|^2}{\sum_{x_i \in c_j} 1}\end{aligned}$$

where $\mathbf{c_j}$ is the cluster center.

We noted that each step of the K-Means algorithm reduces the variance, and that the best clustering has the minimum value of the variance.

Weighted K-Means: The Center of Mass

When point $\mathbf{p_i}$ has weight $\mathbf{w_i}$, the formula for the center of mass is:

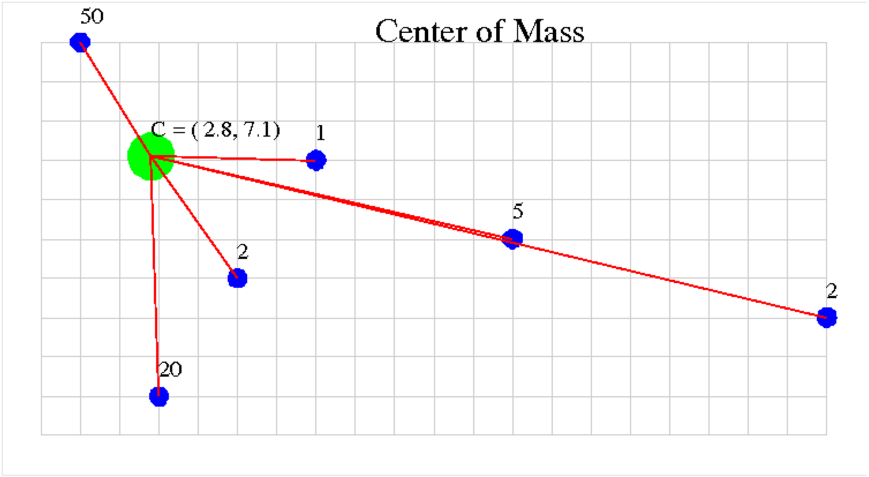
$$\mathrm{c_j} = \frac{\sum_{x_i \in c_j} w_i\, x_i}{\sum_{x_i \in c_j} w_i}$$

and cluster center $\mathbf{c_j}$ is the **center of mass** of its cluster.

The location of the center of mass can be **anywhere** within the convex hull of the set of data points.

Similarly, if each point $\mathbf{p_i}$ is connected to the center of mass, and given weight $\mathbf{w_i}$, this object will also be perfectly balanced.

Weighted K-Means: The Center of Mass



Weighted K-Means: Center of Mass Minimization

The center of mass of a set of weighted points minimizes the weighted variance. For one cluster with data points **x** and weights **w**, the variance with respect to an arbitrary point **c** is

$$\mathrm{var}(\mathrm{x}, \mathrm{w}, \mathrm{c}) = \frac{\sum w_i \|x_i - c\|^2}{\sum w_i}$$

and this is minimized if we take **c** to be the center of mass.

PROOF: Solve $\frac{\partial \mathrm{var}(x,w,c)}{\partial c} = 0$.

Weighted K-Means: Weighted Cluster Variance

For weighted clustering, we define the *weighted cluster variance* as:

$$\begin{aligned}\mathrm{var}(\mathrm{x},\mathrm{w},\mathrm{c}) &= \sum_{j=1}^{k} \mathrm{var}(\mathrm{x},\mathrm{w},\mathrm{c_j}) \\ &= \sum_{j=1}^{k} \frac{\sum_{x_i \in c_j} w_i \|x_i - c_j\|^2}{\sum_{x_i \in c_j} w_i}\end{aligned}$$

where $\mathbf{c_j}$ is the cluster center.

Each step of the weighted K-Means algorithm reduces the weighted cluster variance, and the best clustering minimizes this quantity.

Weighted and Continuous Clustering

WKMeans Algorithm

The weighted K-Means algorithm is very similar to the K-Means algorithm.

Changes occur mainly in two places:

- **update_centers**, we use centers of mass instead of centroid;
- **variance**, where we compute a sum of weighted variances.

WKMeans Algorithm: Main Program

```
function [ c, ptoc ] = wkm ( dim, n, p, w, k )

%% WKM carries out the weighted K-Means algorithm.
%

%
%  Initialize the cluster centers.
%
  c = kmeans_initialize ( dim, n, p, k );
%
%  Repeatedly update clusters and centers til no change.
%
  v = -1;
  while ( 1 )

    ptoc = kmeans_update_clusters ( dim, n, p, k, c );

    c = wkmeans_update_centers ( dim, n, p, w, k, ptoc );

    v_old = v;
    v = wkmeans_variance ( dim, n, p, w, k, c, ptoc );

    if ( v == v_old )
      break
    end

  end

  return
end
```

WKMeans Algorithm: Update the Centers

```
function c = wkmeans_update_centers ( dim, n, p, w, k, ptoc )

%% WKMEANS_UPDATE_CENTERS resets the cluster centers to the weighted data averages.
%
  wp(1:dim,1:n) = p(1:dim,1:n) * w(1:n);

  for j = 1 : k
    index = find ( ptoc(1:n) == j );
    c(1:dim,j) = sum ( wp(1:dim,index), 2 ) / sum ( w(index) );
  end

  return
end
```

WKMeans Algorithm: Compute the Variance

```
function v = wkmeans_variance ( dim, n, p, k, c, ptoc )

%% WKMEANS_VARIANCE computes the variance of the weighted K-means clustering.
%
  pmc(1:dim,1:n) = p(1:dim,1:n) - c(1:dim,ptoc(1:n));

  v = 0.0;
  for j = 1 : k
    index = find ( ptoc(1:n) == j );
    wpmc(index) = w(index) .* ( norm ( pmc(1:dim,index) ) )^2;
    var(j) = sum ( wpmc(index) ) / sum ( w(index) );
    v = v + var(j);
  end

  return
end
```

Weighted and Continuous Clustering

Continuous Data

So far, our data set **X** has been a set of **N** objects in **D**-dimensional space. Whether the objects are actually geometrical, we can think of them as points in that space.

It's natural to ask whether it's essential that **X** is a finite set; in particular, we'd be interested in knowing whether we can cluster a set **X** that comprises all the points in a geometric region, perhaps a square, or a circle or perhaps the surface of a sphere.

It may seem like an awfully big jump, but the idea is to take what we have done with finite sets and see how far we can extend it.

Continuous Data

Ignore for a moment the question of *how* we can do the clustering.

Instead, let's ask, *What would be the result of clustering points in a square?*.

We've already seen that clustering produces a set of special center points **C** with the property that each data point is close to one of the centers.

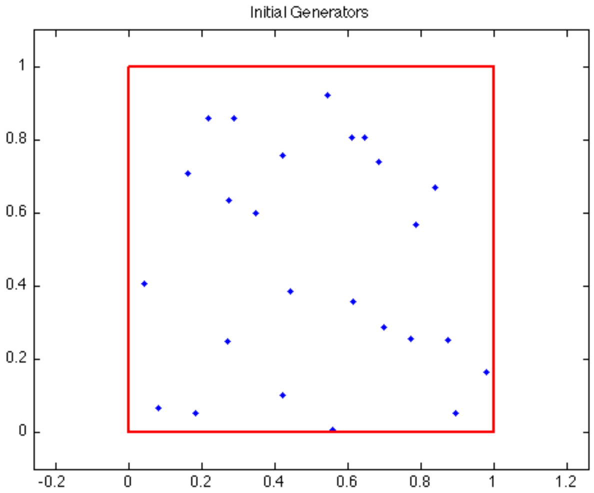
This idea makes perfectly good sense for points in a square.

Continuous Data

So how does our algorithm begin?

We first have to initialize **C** with some random values. Well, that's easy if the region is a square. For other regions, we would clearly need to be able to produce **K** random points that lie in the region. So keep in mind that we have to be able to *sample* the region.

Continuous Data: Initial Centers



Continuous Data - Can't List the Cluster Assignments

Our next step was to assign each data point to the nearest $\mathbf{c_j}$ in $\mathbf{C}$.

In the discrete case, we recorded this information in a vector **ptoc**, and used that information several times. But if $\mathbf{X}$ is infinite, we can't store the cluster map in a vector.

So no matter how many times we want to know which cluster some point $\mathbf{x_i}$ belongs to, we have to figure the answer out from scratch.

So keep in mind that it will be expensive to determine cluster membership, and that we can't save the results for later reference.

Continuous Data - Update Centers by Averaging

Next we averaged the $\mathbf{x_i}$ belonging to each $\mathbf{c_j}$.

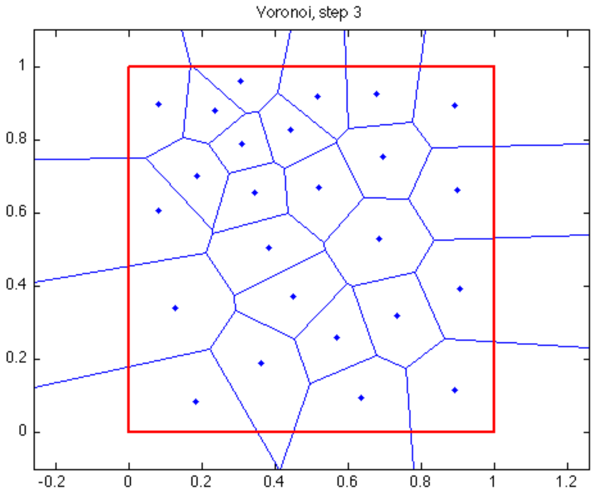
$$\mathrm{c_j} = \frac{\sum_{x_i \in c_j} x_i}{\sum_{x_i \in c_j} 1}$$

I wrote the discrete average that way to suggest how we can replace sums by integrals:

$$\mathrm{c_j} = \frac{\int_{x_i \in c_j} x_i \, dx}{\int_{x_i \in c_j} 1 \, dx}$$

This calculation is harder than it looks, because we are integrating over an irregular region. (We will come back to this point!)

Continuous Clusters - Example of Centers and Clusters



Continuous Data - The Variance

Finally, knowing the new $\mathbf{c_j}$, we computed the variance.

$$\begin{aligned}\mathrm{var(x,c)} &= \sum_{j=1}^{k} \mathrm{var(x,c_j)}\\ &= \sum_{j=1}^{k} \frac{\sum_{x_i \in c_j} \|x_i - c_j\|^2}{\sum_{x_i \in c_j} 1}\end{aligned}$$

This becomes:

$$\begin{aligned}\mathrm{var(x,c)} &= \sum_{j=1}^{k} \mathrm{var(x,c_j)}\\ &= \sum_{j=1}^{k} \frac{\int_{x_i \in c_j} \|x_i - c_j\|^2 \, dx}{\int_{x_i \in c_j} 1 \, dx}\end{aligned}$$

Continuous Data - Time for Details

Now, what we have done is to think about our new problem with a continuous $\mathbf{X}$ and decide that everything we did in the discrete case can pretty much be copied or slightly adjusted, and the next thing you know we'll be solving these problems too!

But it's time for some more careful analysis. In particular, when we replaced sums by integrals, we went from a simple task to a very hard one!

Continuous Data - Approximating an Integral

To focus attention, let's ask how we could compute an integral like

$$\int_{x_i \in c_j} 1 \, dx$$

which represents the area or volume or hypervolume of the points that belong to the cluster centered at $\mathbf{c_j}$. If we were working in a 2D region, this cluster would probably be a convex polygon, but we have no idea of how many sides it has or where the vertices are!

If we are willing to live with an *estimate* of this value, then there is a reasonable approach that will work for any shape, and any number of dimensions, called **Monte Carlo sampling**.

Continuous Data - Approximating an Integral

The integral

$$\int_{x_i \in c_j} 1 \, dx$$

represents, in the 2D case, the area of the cluster. Let's suppose we're working in the unit square. The cluster associated with $\mathbf{c_j}$ is a polygonal portion of the unit square.

So we compute a large number **S** of sample points (we said we would have to know how to do that). We count the number of sample points that are part of $\mathbf{c_j}$'s cluster to estimate for the relative area of that cluster. Since the total area is 1 in this case, we actually are estimating the area itself.

Using this same idea, we can estimate the area of each cluster.

Continuous Data - Approximating an Integral

OK, but how do we approximate integrals like

$$\int_{x_i \in c_j} \|x_i - c_j\|^2 \, dx$$

which we need for the variance?

Again, we can use the same set of **S** sample points. Each sample point $\mathbf{s_j}$ that lies in $\mathbf{c_j}$'s cluster will contribute to our estimate. The integral estimate is simply

$$\int_{x_i \in c_j} \|x_i - c_j\|^2 \, dx \approx \mathbf{Area}(c_j) \frac{\sum_{s_i \in c_j} \|s_i - c_j\|^2}{\sum_{s_i \in c_j} 1}$$

VirginiaTech

Continuous Data: A CKMeans Program

Our MATLAB program for this problem is similar to the others, except that we can't compute the **ptoc** array. Instead, we rename our current centers **c_old**, we use sampling to find all the points nearest each centroid, and average those to get the new set of centers **c**.

Then we estimate the variance to see if we should keep working.

These calculations are all estimated using sampling, so we may find that increasing the size of the sampling set **S** will improve our answers.

Continuous Data - Main Program

```
function c = ckm ( dim, n, p, w, k )

%% CKM carries out the continuous K-Means algorithm.
%

%
%  Initialize the cluster centers.
%
  c = ckmeans_initialize ( dim, n, p, k );
%
%  Repeatedly update clusters and centers til no change.
%
  v = -1;
  while ( 1 )

    c_old = c;
    c = ckmeans_update_centers ( dim, n, p, w, k, c_old );

    v_old = v;
    v = ckmeans_variance ( dim, n, p, w, k, c, ptoc );

    if ( v == v_old )
      break
    end

  end

  return
end
```

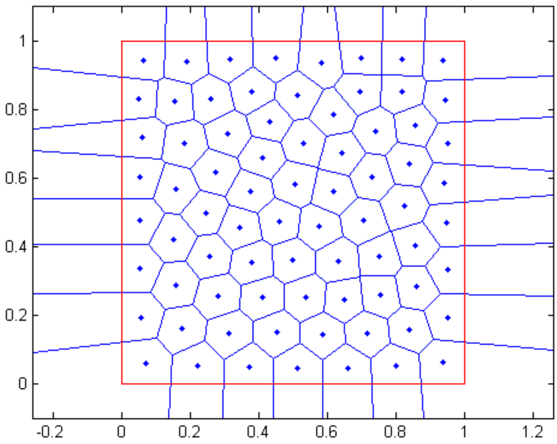
Continuous Data - Main Program

Despite all the approximation that goes on, the results can have some surprising structure.

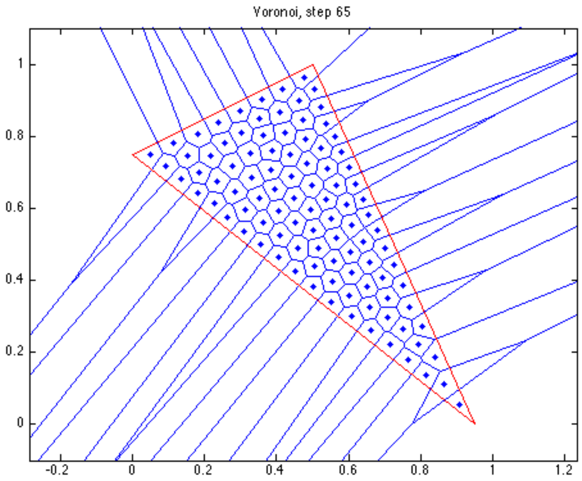
We only talked about the unit square, but you can do this kind of process on any region for which you can produce sample points.

Often, an engineer or scientist needs to break down a region into simple subregions for analysis. This method gives you polygonal regions (in 2D). The subdivision, which assigns each point to the nearest center, is known as the Voronoi diagram.

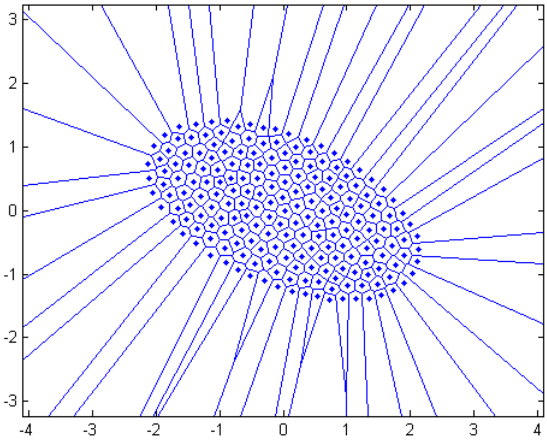
Continuous Data: On a Square



Continuous Data: In a Triangle



Continuous Data: In an Ellipse



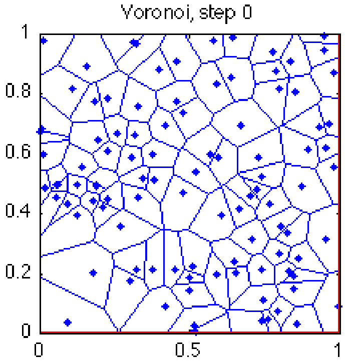
Continuous Data: Delaunay Triangulation

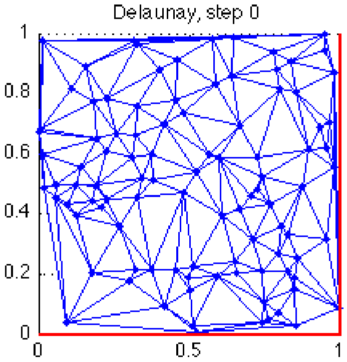
Often you want your original region to be broken up into triangular subregions.

This is especially true in engineering problems which use finite elements, and in graphics applications which expect a regular mesh of triangles.

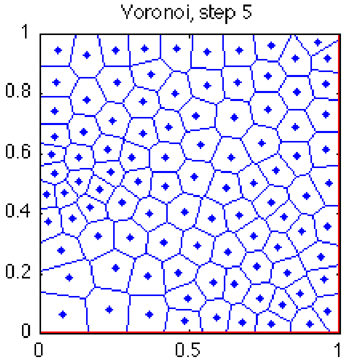
Because our continuous K-Means algorithm is giving us a Voronoi Diagram, we can draw lines that connect the centers to create the dual graph, which is known as the Delaunay triangulation. This will have the correct structure.

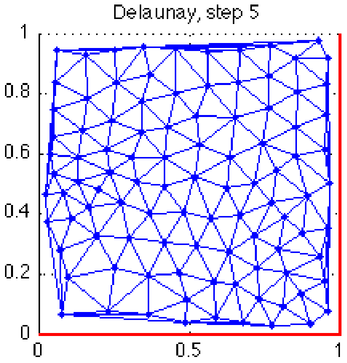
Continuous Data: Voronoi/Delaunay Step 1



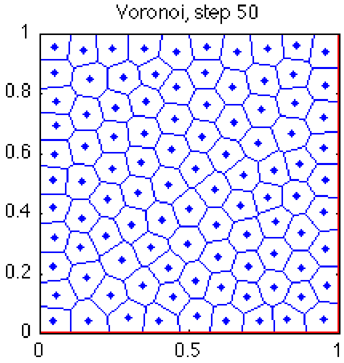


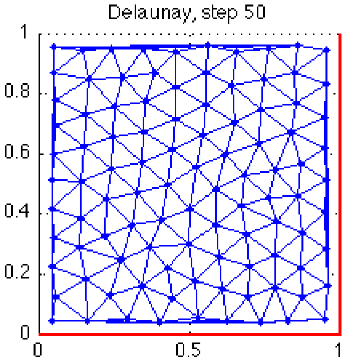
Continuous Data: Voronoi/Delaunay Step 5





Continuous Data: Voronoi/Delaunay Step 50





Weighted and Continuous Clustering

- Weighted K-Means
- A Weighted K-Means Algorithm
- K-Means for Continuous Data
- **Weighted K-Means for Continuous Data**

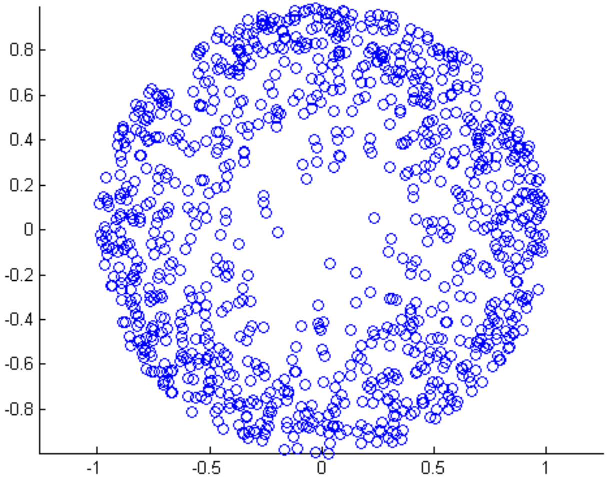
Weighted Continuous Data

Just as in the discrete case, it is often the case that some parts of the data are much more important that others.

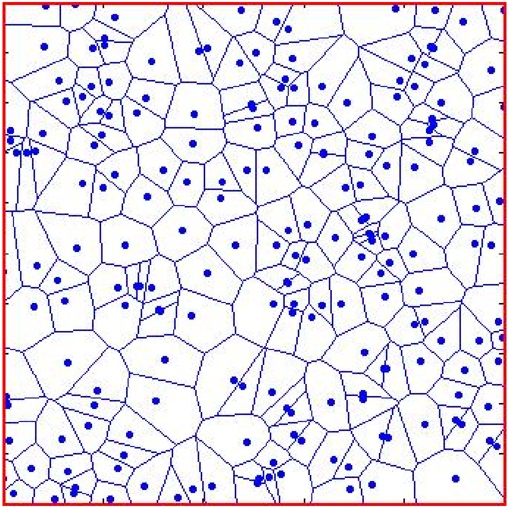
If an engineer is using the continuous K-Means algorithm to form a mesh of an aircraft, it may be important that there be lots of points near places where cracks are likely to form, such as where the wings connect to the body.

It is easy to add weights to the continuous case. Instead of a weight vector, we have a weight function $\mathbf{w}(\mathbf{x})$, which shows up in the integrals.

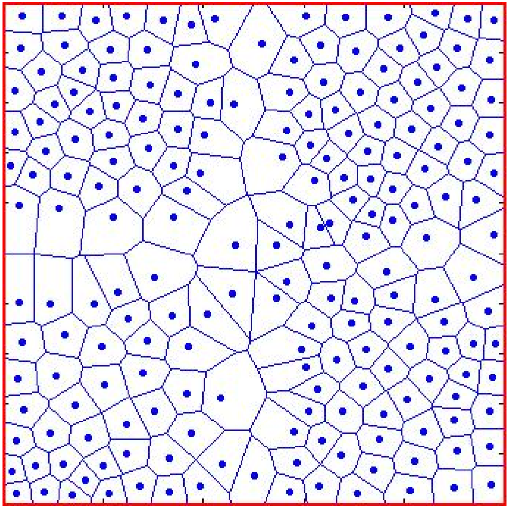
Weighted Continuous Data: A Circle



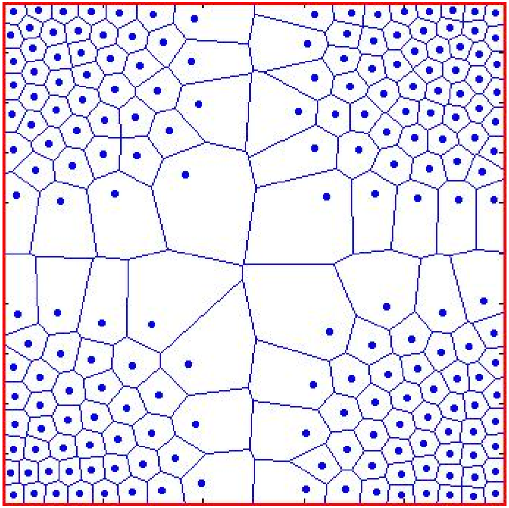
Weighted Continuous Data: A Square, Step 1



Weighted Continuous Data: A Square, Step 5



Weighted Continuous Data: A Square, Step 50



Conclusion

The K-Means algorithm can be extended to use weights, and it can be extended to the case where the objects we are grouping are actually regions in space.

Weighting allows for a more sophisticated grouping, which takes into account that some things are more important, or more valuable, or have more information.

Clustering continuous regions of space is only possible when we use approximations. However, it enables us to achieve regular subdivisions of regions even when those regions have unusual shapes, and to deal with weight functions.