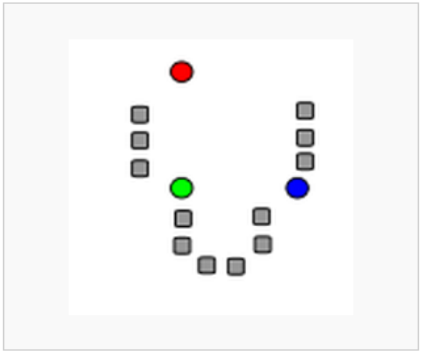
In finance, we often want to treat things as a group. The hope is that the grouping will reveal patterns and reduce noise. For example, we know for a fact that the returns of 7,000 stocks do not represent 7,000 orthogonal factors. The number of factors driving stock returns is much, much smaller than that, though the exact number of real factors is a point of some controversy.

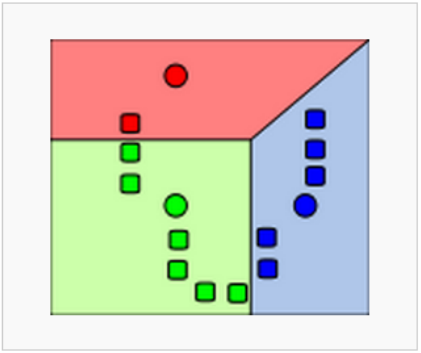
Techniques such as principal components analysis – which is frequently used to attempt to extract the factors driving stock prices – are beyond the scope of this class. You will spend some time working with PCA and other approaches later in the math finance program. In this class, we want to focus on problems that exercise your programming muscles.

Fortunately, there is a computationally interesting yet relatively simple clustering technique that we can use to demonstrate the problem of grouping things. It's called Lloyd's Algorithm or the K-means algorithm. The following is a graphical demonstration of the standard Lloyd's Algorithm at work.

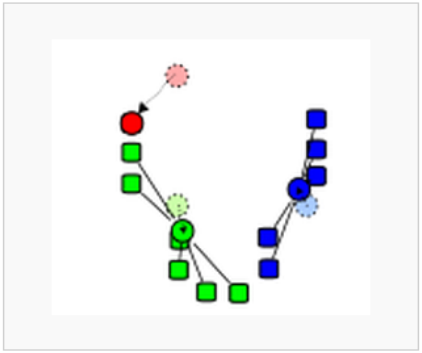
Demonstration of the standard algorithm



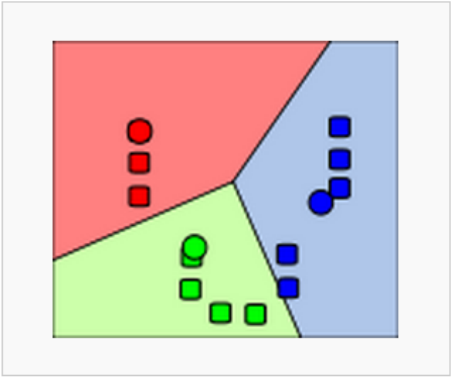
1) *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3) The centroid of each of the *k* clusters becomes the new mean.



4) Steps 2 and 3 are repeated until convergence has been reached.

While the above clustering problem is shown in two dimensions, any number of dimensions can be used. We can compute distances in two dimensions:

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

..or three dimensions:

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$$

..or any number of dimensions.

If we were doing this with a time series of stock returns, each return would represent a value in one dimension, and each time series would be a point in n dimensions, where n is the number of returns in that series.

But we're going to leave this bigger problem alone and focus on a much smaller problem, a toy problem. As explained in class, though the solution of the toy problem will be well known to us, it will not be easy to get a computer to solve it.

The Toy Problem

Suppose a city 99 blocks wide by 99 blocks long. A person stands on every corner, 10,000 people in all. In other words, there's a person at {0,0}, a person at {0,1}, and so on. There is a person at {99,99}, the top right corner of the map, a person at {0,99}, the top left corner of the map, and a person at {99,0}, the bottom right corner of the map. 10,000 people in all, evenly distributed throughout our toy city. (By the way, this area is about twice the size of Manhattan.)

The problem we want to solve is this: we want to group these people into clusters of 20, or 500 clusters in all. The solution to this problem is a list of the centers of these clusters, their geometric *centroids*.

This problem is called NP hard. It's a highly combinatorial problem and brute force solutions are going to fail miserably. For example, suppose we were finding just two cluster centroids. There are 100 possible x coordinates and 100 possible y coordinates for each of two cluster centroids, or 100 x 100 x 100 x 100 = 100 million possible ways to place the centroids. But we are not working with two centroids. We are working with 500. Clearly, a brute force search just won't work.

Intuitively, we know the solution: we would put one cluster centroid at the center of every square with the following width and height in blocks:

$$100/\sqrt{(500)} \;=\; \sim 4.5$$

However, to actually do this, we will use Lloyd's algorithm. The solution will not be exact. Lloyd's Algorithm is a heuristic which will converge to solutions that may not be globally optimal. However, it will get us pretty close.

The Objectives

Any algorithm that can be graphically described in 4 slides (as above) is going to be trivial to implement, so the solution – while it should be correct – is not the main objective of this homework. The main objective of this homework is to apply the themes and strategies that were explained to you in the last lecture.

- Your solution should be elegant in its object-oriented design and should lend itself to easy and thorough unit testing using Junit.
- This testing is part of the deliverables of your homework assignment.
- You should build your classes to be reusable. Though the problem is simple, there are definitely reusable components that are part of the solution.
- You should come up with a metric that allows you to measure the quality of the solution as well as the quality of the initial state. Your code should be readable and well documented.

The Twist

As you can see from the graphical representation of Lloyd's Algorithm, it does not group points in clusters of fixed size. But what if we want that? What if we demand that each cluster has 20 people? The problem becomes much more difficult. To see how much more difficult, you will present two solutions. One solution will use the standard Lloyd's approach – points seeking the nearest cluster – while the second solution will use a modified approach – clusters seeking the nearest 20 points.

In other words, in Lloyd's Algorithm, you will iterate over every point – each representing a person – and, for that point, will find the nearest cluster centroid. In the modified approach, you will iterate over cluster centroids and, for each, will find the nearest 20 points.

If you write your code correctly and use all of the principles of good object oriented design, this modification should be easy and should not require rewriting much of your code.