K-Means Clustering

John Burkardt (ARC/ICAM)
Virginia Tech

.....

Math/CS 4414:
"K-Means Clustering"
http://people.sc.fsu.edu/~jburkardt/presentations/clustering_kmeans.pdf

.....

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

21 September 2009

VirginiaTech

K-Means Clustering

- **Overview**
- Clustering
- The K-Means Algorithm
- Running the Program

VirginiaTech

Overview: K-Means Clustering

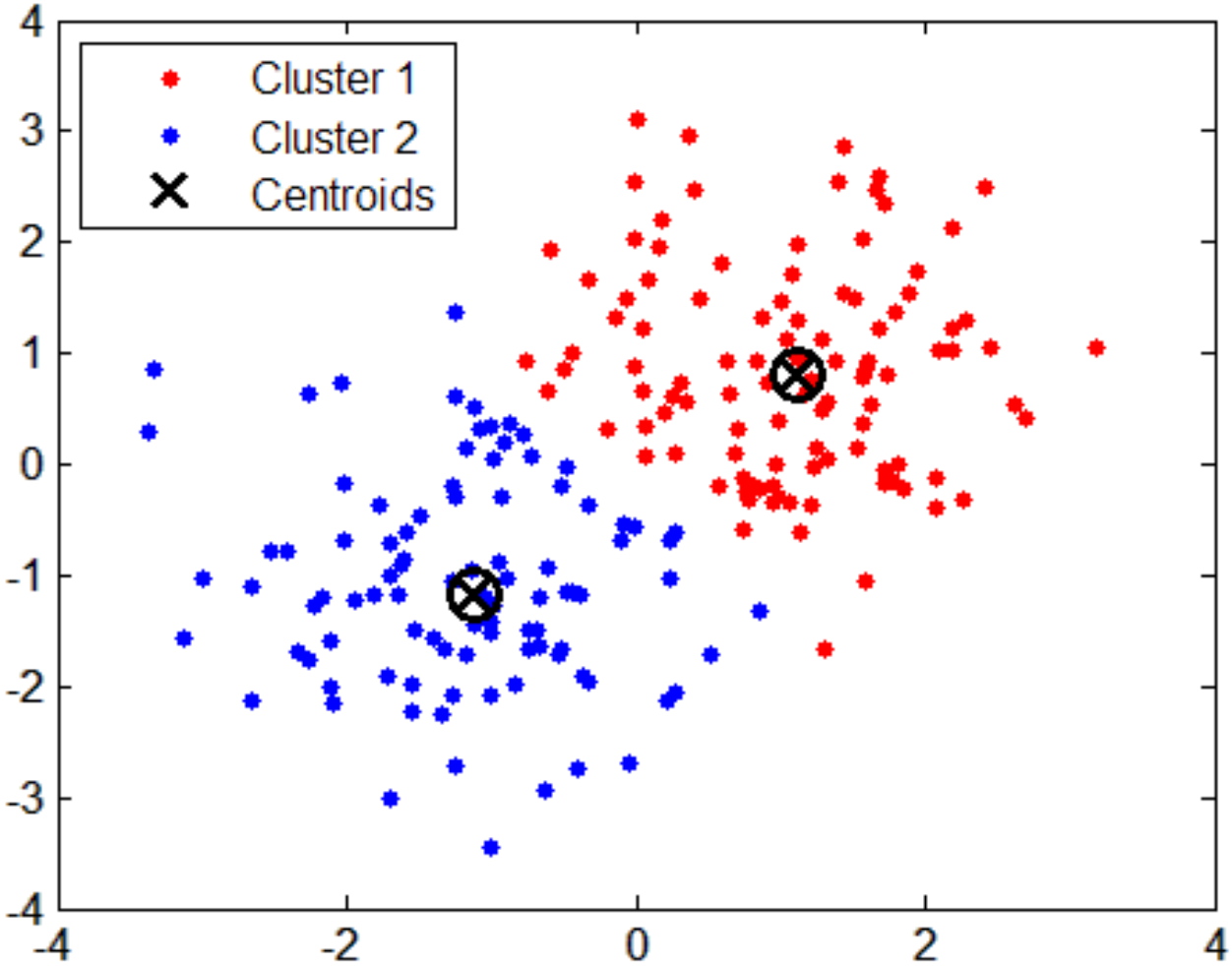
In the previous lecture, we considered a kind of hierarchical clustering called *single linkage clustering*. This was useful because we thought our data had a kind of family tree relationship, and single linkage clustering is one way to discover and display that relationship if it is there.

Today we will look at a different clustering tool called **K-Means**. The fundamental idea is that we are going to look for **K** average or mean values, about which the data can be clustered. Our goal now is perhaps no so much to find a family history as to simply break the data up into **K** groups.

We might be doing this in the hope that each group can be "explained" by some common parameter value.

Or we might not be looking for understanding - instead we might simply want to compress the data.

Overview: An Example of K-Means Clustering



VirginiaTech

Overview: Examples of Data for Clustering

The data that K-Means works with must be numerical. Each data object must be describable in terms of numerical coordinates.

We can imagine that these coordinates represent a spatial position. A surprising number of things can be described this way, including

- the weekly position of one song in the Top 40 ratings;
- the election results, by party, in one particular town;
- the medical test results for one patient in a study;
- the RGB coordinates of one color used in an image

VirginiaTech

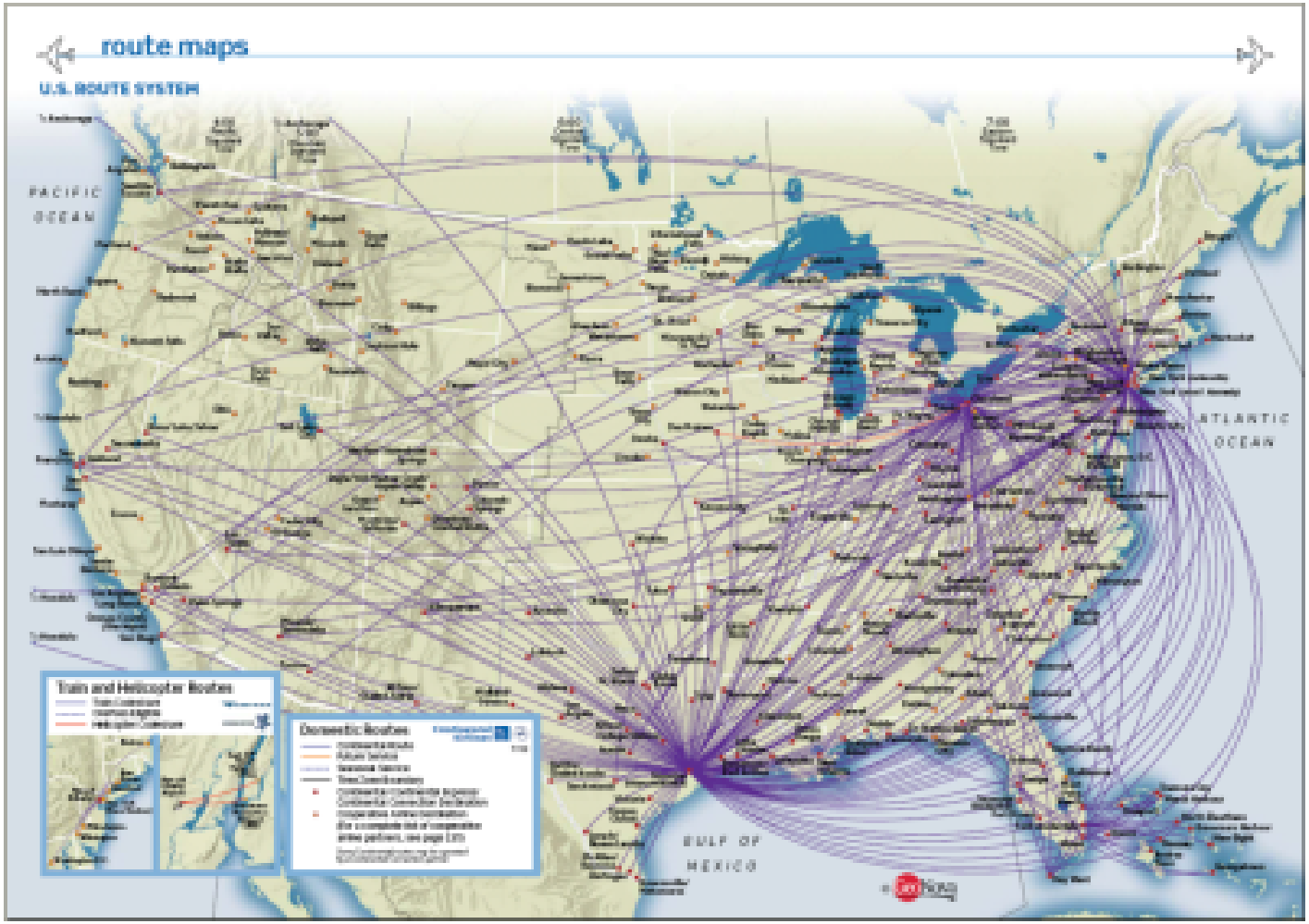
Overview: Examples of Data for Clustering

We aren't looking for a theory about how these objects are created or why they are different. We are instead trying to get a representative sample of the data, so that we can see the most typical behaviors and then try to explain patterns that we see:

- do most hit songs vanish after 10 weeks? do songs by new artists tend to be short-lived?
- how do towns differ, based on differences in the election results?
- if some patients in the study got very sick afterwards, are there some test results that would have predicted this?
- if we can only use 8 colors in an image, what is a choice that would be representative?

VirginiaTech

Overview: Clustering Geometric Data



Overview: Clustering Geometric Data

Sometimes the data for K-Means really is spatial, and in that case, we can understand a little better what it is trying to do.

We can use K-means clustering to decide where to locate the **K** "hubs" of an airline so that they are well spaced around the country, and minimize the total distance to all the local airports.

A better approach to this problem, of course, would take into account the fact that some airports are much busier than others. To do that requires a **weighted K-means** clustering, which we may talk about later.

VirginiaTech

Overview: Variance

There is a statistical concept called *variance* that we will find useful for our clustering work. The classical definition of variance measures the squared difference between each data item and the mean value, and takes the average.

$$\text{ave}(\text{x}) = \frac{1}{n}\sum_{1}^{n} x_i$$

$$\text{var}(\text{x}) = \frac{1}{n}\sum_{1}^{n} (x_i - \text{ave}(\text{x}))^2$$

If a set of data has a small variance, most of the data is close to the average.

VirginiaTech

Overview: Variance

For clustering, we will also want to measure the closeness of data to an average. However, instead of a single average, we will divide our data into clusters, each of which will have its own average. We won't bother normalizing by the number of data items, and we will have to adjust for the fact that our data is **D**-dimensional.

This gives us something we can call *cluster variance*:

$$\mathrm{var}(\mathrm{x}) = \sum_{1}^{n} \|x_i - \mathrm{ave}(\mathrm{x_i})\|^2$$

Here **ave**($\mathbf{x_i}$) represents the *cluster average* (although when we get around to explaining this, we will actually call it the *cluster center*).

K-Means Clustering

- Overview
- **Clustering**
- The K-Means Algorithm
- Running the Program

VirginiaTech

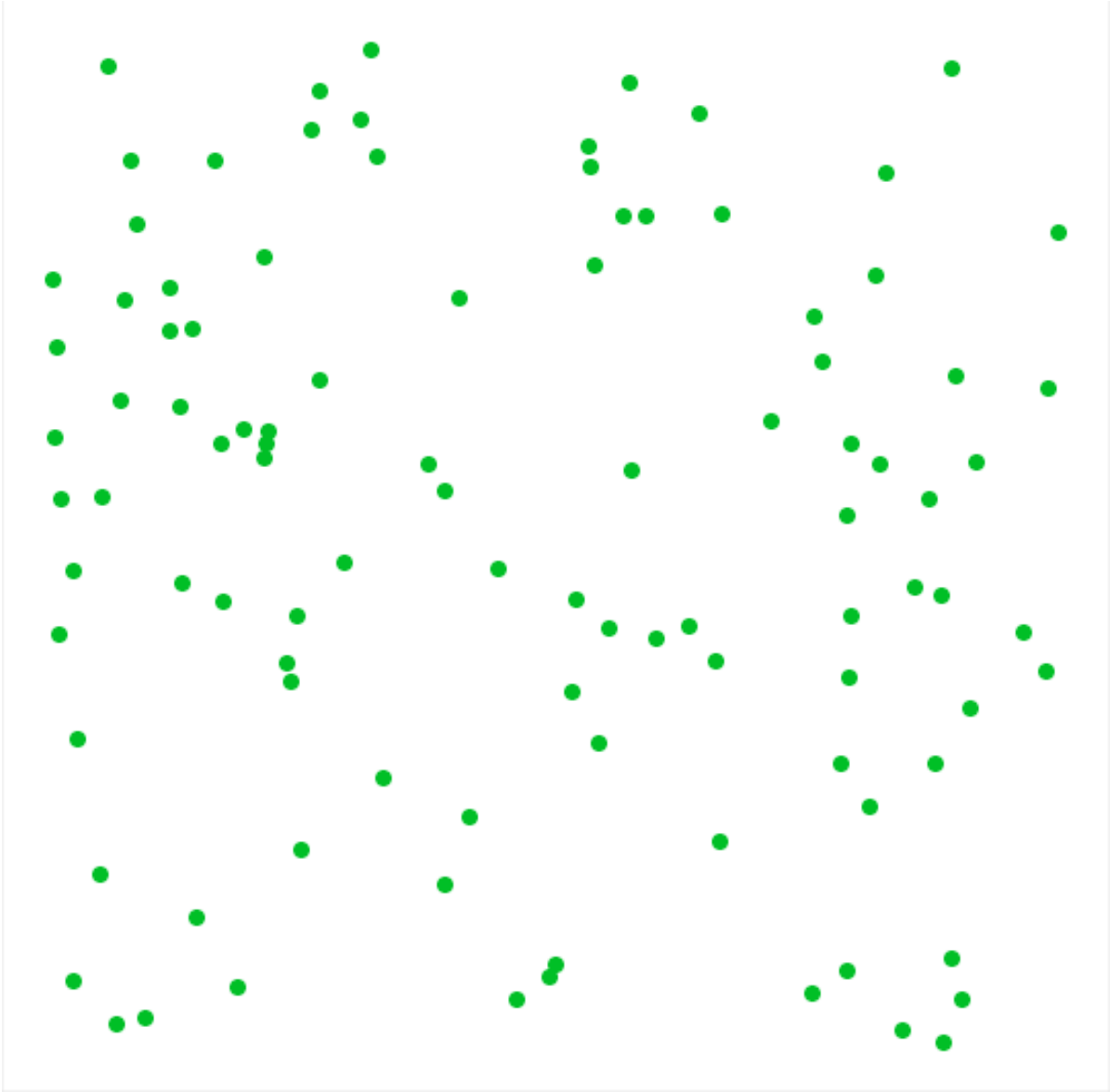
Clustering: Problem Statement

We suppose that we are given a set of **N** items **P**, with a typical entry being $\mathbf{p_i}$, and that each $\mathbf{p_i}$ is a vector of **D** numbers.

We can think of each $\mathbf{p_i}$ as being a point in a **D**-dimensional space.

VirginiaTech

Clustering: P = 100 Points in 2D



VirginiaTech

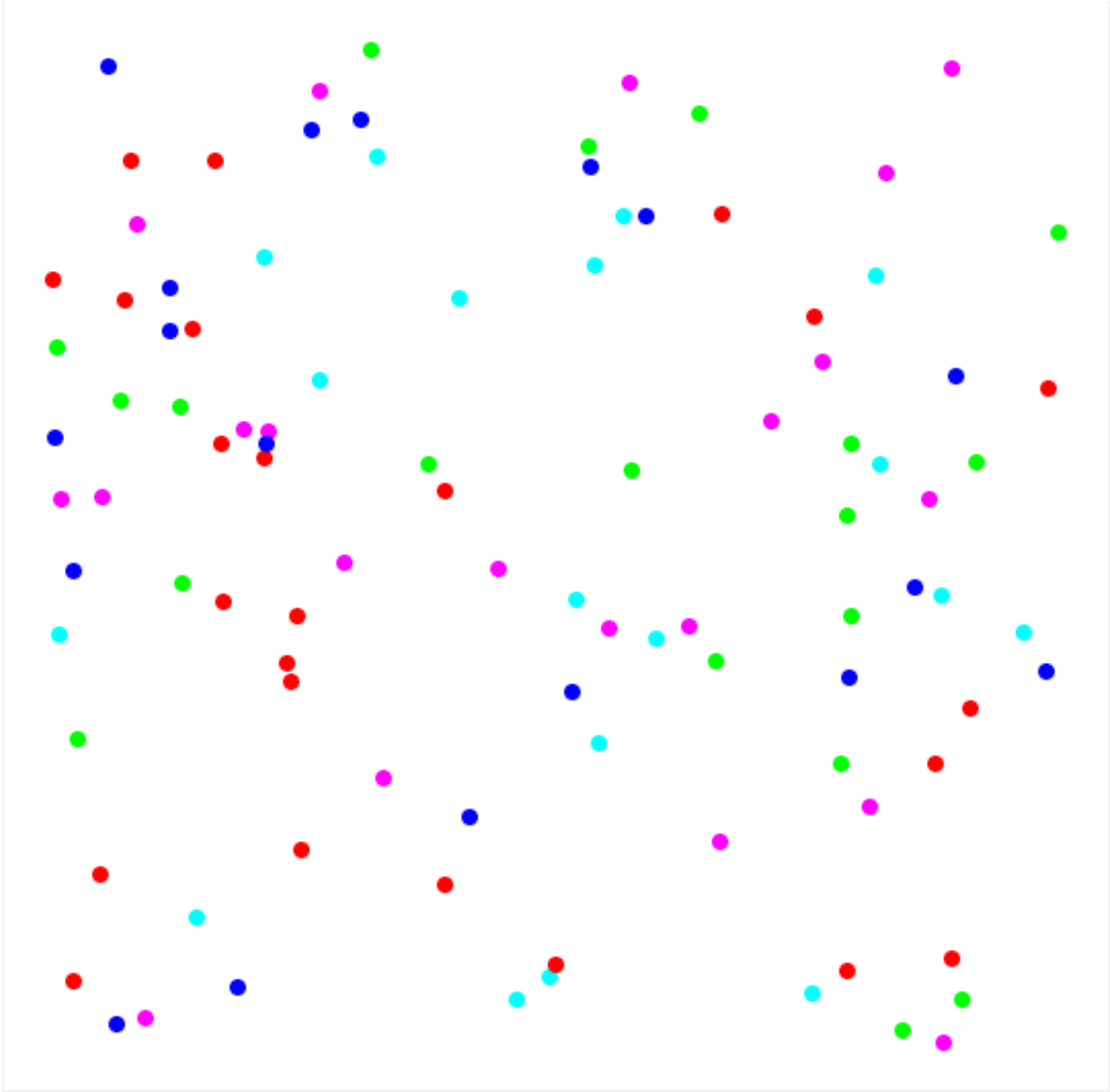
Clustering: Problem Statement

Our desire is to break the data up into **K** clusters. We can do this in any way we like. We imagine we have a function or rule, which we symbolize by **PtoC**() so that if $j = \mathbf{PtoC}(\mathbf{p_i})$, the point $\mathbf{p_i}$ belongs to cluster **j**.

We can certainly break the points up into **K** clusters. That's easy!

VirginiaTech

Clustering: P = 100 Points in 2D, K = 5 Random Clusters



VirginiaTech

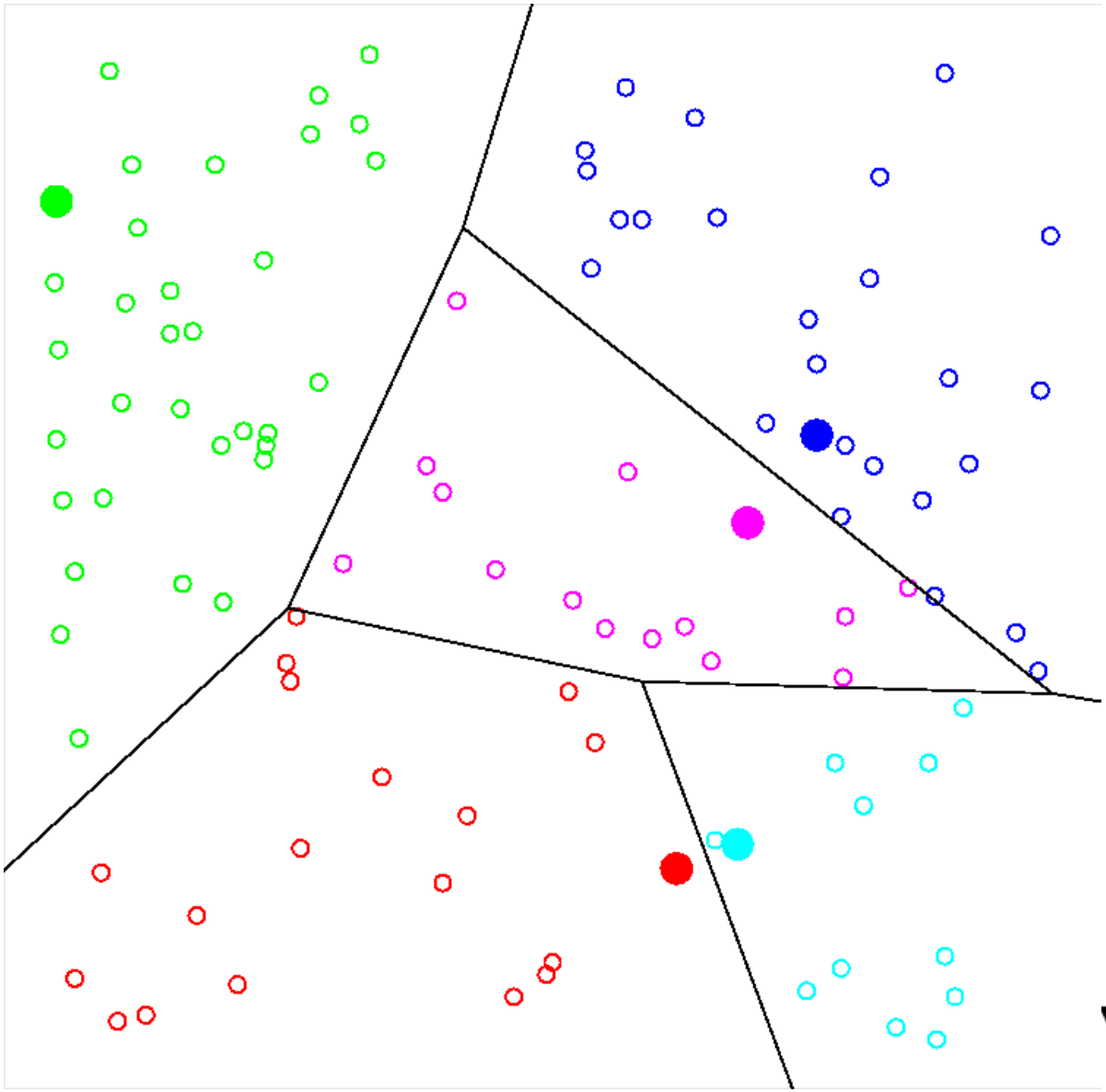
Clustering: Using Random Centers is Not Great

A random clustering doesn't use any of the geometric information to try to group our data. A better method chooses a set **C** of **K** "cluster centers", one for each cluster, and then associates a point $\mathbf{p_i}$ with the cluster whose center $\mathbf{c_j}$ is nearest.

Unfortunately, this doesn't tell us how to pick the values **C**, and in fact, often we're doing clustering precisely because we are seeking a good set of **C** values!

VirginiaTech

Clustering: P = 100 Points in 2D, K = 5 Random Centers



VirginiaTech

Clustering: Use the Data to Move the Centers!

Even though the clustering induced by our random centers wasn't great, it did divide the data. In fact, the oddest thing about the clusters now is that the centers aren't actually in the center.

So we got our grouping (good) but the cluster centers aren't actually good representatives of the clusters (bad).

Suppose we move our center points to the actual centers of the clusters. Then it's likely that the centers are more evenly spaced throughout the region, and this will also likely reduce the average distance of cluster points to the center.

VirginiaTech

Clustering: Use the Centers to Move the Data!

Moving the centers to the center is a good idea. It will cause the variance to decrease giving us a better clustering than before. Interestingly, though, some of the points in each cluster now are closer to the center of a different cluster.

Since we want the points to cluster to the nearest center, we can transfer those points to the cluster where they seem to want to go. This will also reduce the variance!

VirginiaTech

Clustering: It's Time For an Iteration

Even if a single point moves to another cluster, this affects the locations of the cluster centers of the cluster it left and the cluster it joined.

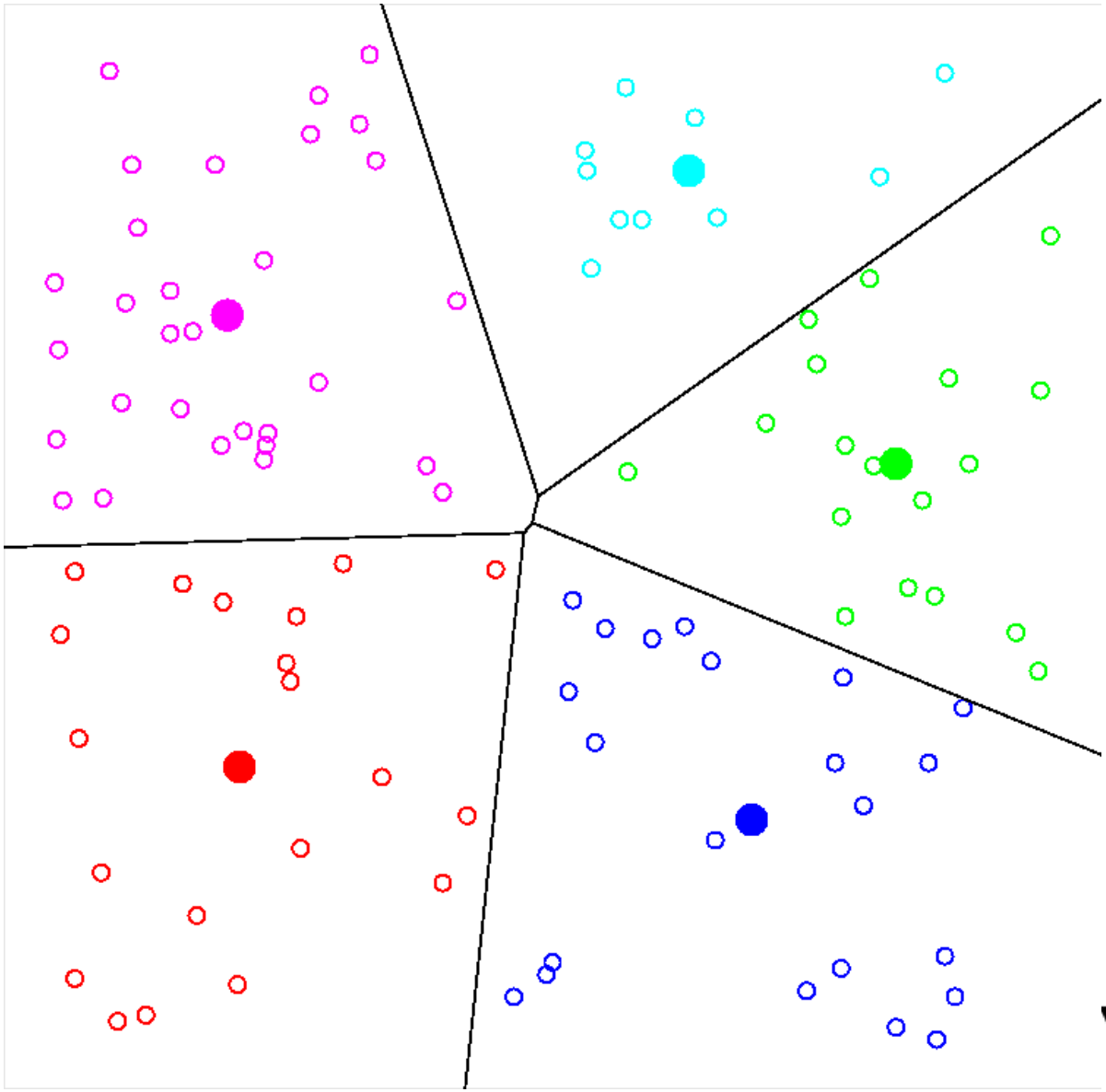
That's because we are now defining the cluster centers to be the averages of all the points in the cluster.

So once again we must update the cluster centers, and this in turn may cause some points to want to transfer.

We clearly will need to carry this out as an iteration. Luckily, each step is guaranteed to reduce the variance. There is a minimum variance, and there are only finitely many possibly clusterings (even though this number is enormous) so the iteration will terminate.

But will it terminate at a clustering with the minimum possible variance?

Clustering: P = 100 Points in 2D, K = 5 K-MEANS



VirginiaTech

K-Means Clustering

- Overview
- Clustering
- **The K-Means Algorithm**
- Running the Program

VirginiaTech

The K-Means Algorithm

Choose initial centers $\mathbf{c_1}, \ldots, \mathbf{c_k}$.
..
While the clusters are changing,
..*Reassign the data points.*
..**For i** = 1, ..., **n**
....Assign data point $\mathbf{p_i}$ to the cluster
....whose center $\mathbf{c_j}$ is closest.
..end
..*Update the cluster centers.*
..**For j** = 1, ..., **k**
....$\mathbf{n_j}$ = number of points in C_j;
....$\mathbf{c_j} = \frac{1}{n_j} \sum_{p_i \in C_j} p_i$
..end
end

The K-Means Algorithm: Initialization

The initialization step is actually important. Because the algorithm can sometimes stop too soon, it's good to be able to retry it with different starting values, perhaps several times.

If appropriate, you can set your centers using the **rand()** function.

If the range of your data is not [0,1], then you can scale the output of the **rand()** function.

You can use MATLAB's **randperm()** function to come up with a random permutation of the **n** data values, and choose the first **k** data values to be your starting centers.

VirginiaTech

The K-Means Algorithm: Initialization

```
function c = kmeans_initialize ( dim, n, p, k )

%% KMEANS_INITIALIZE randomly chooses K data values for cluster centers.
%
%  Get a random permutation of the integers 1 through N.
%
  perm = randperm ( n );
%
%  Use the first K entries of the permutation to select K
%  data values to use as cluster centers.
%
  c(1:dim,1:k) = p(1:dim,perm(1:k));

  return
end
```

The K-Means Algorithm: Update the Clusters

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

%% KMEANS_UPDATE_CLUSTERS assigns data to clusters based on the centers.
%
  for i = 1 : n
    for j = 1 : k
      dist(j) = norm ( p(1:dim,i) - c(1:dim,j) );
    end
    [ dummy, ptoc(i) ] = min ( dist(1:k) );
  end

  return
end
```

xmin = **min**(**x**) returns the minimum value of a vector;
[**xmin**, **i**] = **min**(**x**) returns the minimum and its index;

The K-Means Algorithm: Update the Centers

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

%% KMEANS_UPDATE_CENTERS resets the cluster centers to the data averages.
%
  for j = 1 : k
    index = find ( ptoc(1:n) == j );
    nj = length ( index );
    c(1:dim,j) = sum ( p(1:dim,index), 2 ) / nj;
  end

  return
end
```

index = (**ptoc**(**1** : **n**) == **j**) returns indices where **ptoc** = **j**.
nj = **length**(**index**) tells us how many indices there were.
sum(**p**(**1** : **dim**, **index**), **2**) sums on the second index.

The K-Means Algorithm: Compute the Variance

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

%% KMEANS_VARIANCE computes the variance of the K-means clustering.
%
  v = 0.0;
  for i = 1 : n
    j = ptoc(i);
    v = v + ( norm ( p(1:dim,i) - c(1:dim,j) ) )^2;
  end

  return
end
```

This quantity is similar to the statistical variance, but it is not the same. (We have multiple centers, we don't divide by the number of objects, and we add the **K** variances together.)

VirginiaTech

The K-Means Algorithm: Main Program

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

%% KM carries out the 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 = kmeans_update_centers ( dim, n, p, k, ptoc );

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

    if ( v == v_old )
      break
    end

  end

  return
end
```

The K-Means Algorithm: Overview of the Program

The program is "randomized" because we call **randperm**() for initialization. Each time we call, we will get a new random set of starting centers.

It is possible, when we update the centers, that a particular center will have no data points in its cluster. This will cause this algorithm to break. How can you fix that?

If the algorithm is written correctly, then on each iteration, if any point moves from one cluster to another, the variance will decrease. Therefore, if the variance stops decreasing, this execution of the program is done.

VirginiaTech

K-Means Clustering

- Overview
- Clustering
- The K-Means Algorithm
- **Running the Program**

VirginiaTech

Running the Program: Random Data

The program **km.m** can be used to solve a simple K-Means problem.

To define a random problem set **P** of dimension **D** and size **N** and try the program on it you can simply type the commands:

```
dim = 2;
n = 100;
p = rand ( dim, n );
k = 5;
[ c, ptoc, v_data ] = km ( dim, n, p, k );
```

The extra output argument **v_data** contains the variance after each iteration. We will want to plot that!

VirginiaTech

Running the Program: Observing Local Minimums

Since random data doesn't break into groups naturally, there will be cases where the program returns a "local minimum" (not the best answer). To see this, run the program several times, and look at the *last* entry of **v_data**, which you get by referring to index **end**:

```
[ c, ptoc, v_data ] = km ( dim, n, p, k );
v_data(end)
  ans = 16.3745
[ c, ptoc, v_data ] = km ( dim, n, p, k );
v_data(end)
  ans = 15.5957
[ c, ptoc, v_data ] = km ( dim, n, p, k );
v_data(end)
  ans = 16.5975
```

VirginiaTech

Running the Program: Plotting the Variance

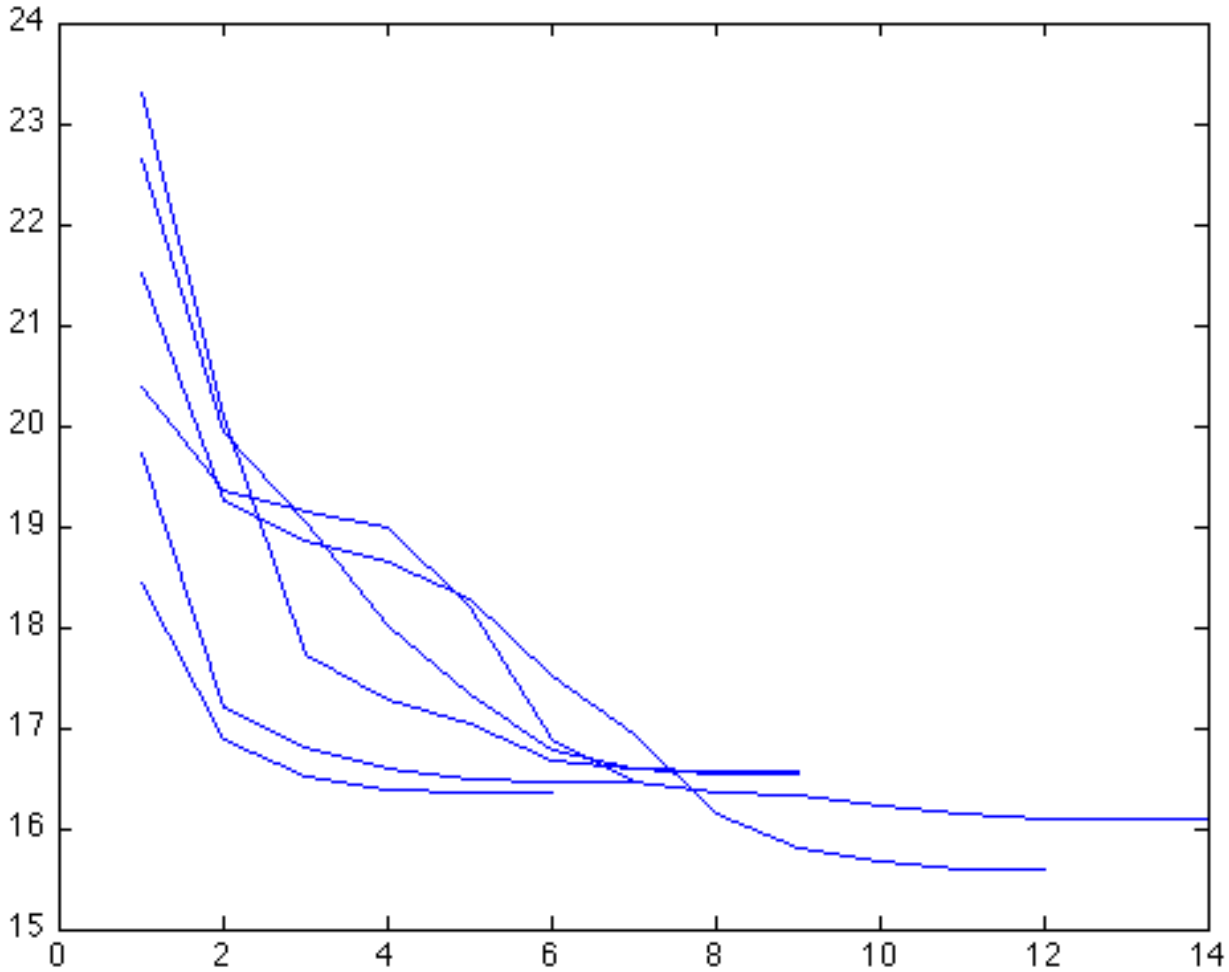
Another way to get a feel for what is going on is to plot the whole **v_data** vector for several runs. To do this, you need to use the **hold on** command so that each new plot is added to the current one.

```
[ c, ptoc, v_data ] = km ( dim, n, p, k );
plot ( v_data );
hold on
[ c, ptoc, v_data ] = km ( dim, n, p, k );
plot ( v_data );
[ c, ptoc, v_data ] = km ( dim, n, p, k );
plot ( v_data );
```

Type **hold off** when you are done!

VirginiaTech

Running the Program: Multiple Variance Plots



Running the Program: The Scores Datafile

The file *scores.txt* contains the scores for reading and arithmetic tests at the 4th and 6th grade levels, at 25 area elementary schools.

The text begins like this:

```
2.7   3.2   4.5   4.8
3.9   3.8   5.9   6.2
4.8   4.1   6.8   5.5
3.1   3.5   4.3   4.6
3.4   3.7   5.1   5.6
3.1   3.4   4.1   4.7
4.6   4.4   6.6   6.1
3.1   3.3   4.0   4.9
...
(25 lines total)
```

VirginiaTech

Running the Program: Data From a File

To cluster the data in *scores.txt*, we need to get a copy of the file in our current MATLAB directory, and read the data using the **load** command:

```
p = load ( 'scores.txt' );
```

Each line of the data file represents one of our "points". But when we store data in MATLAB, we want each column of the array to be a point. So we have to transpose the data that we just read, before we can run the program.

```
p = p';
[ dim, n ] = size ( p );
k = 3;
[ c, ptoc, v_data ] = km ( dim, n, p, k );
```

Running the Program: ASSIGNMENT

Copy the program **km.m** and the dataset *scores.txt* from Scholar. They will be available in the **Clustering** Resource.

Read the dataset into MATLAB as demonstrated on the previous slide. Transpose the data so it is 4x25 in dimension. Then run the **km** program, using **K**=3 clusters. Record the final value of **v_data** that you get.

Run the program five times.

Turn in:

- EITHER a list of the final variance for each run (5 numbers);
- OR one plot that shows all 5 **v_data** curves.

Submit this to SCHOLAR, or turn it in to me, by class time, Friday, 25 September!

VirginiaTech

Running the Program: Background

The material on the K-Means algorithm is based on Chapter 11 of the textbook, *"Classified Information: The Data Clustering Problem"*.

A copy of these slides is available online through Scholar, in the **Clustering** Resource. The file is called *clustering_kmeans.pdf*.

You can also get a copy of these slides from my web page: *http://people.sc.fsu.edu/~burkardt/presentations/clustering_kmeans.pdf*

VirginiaTech