Chapter 5

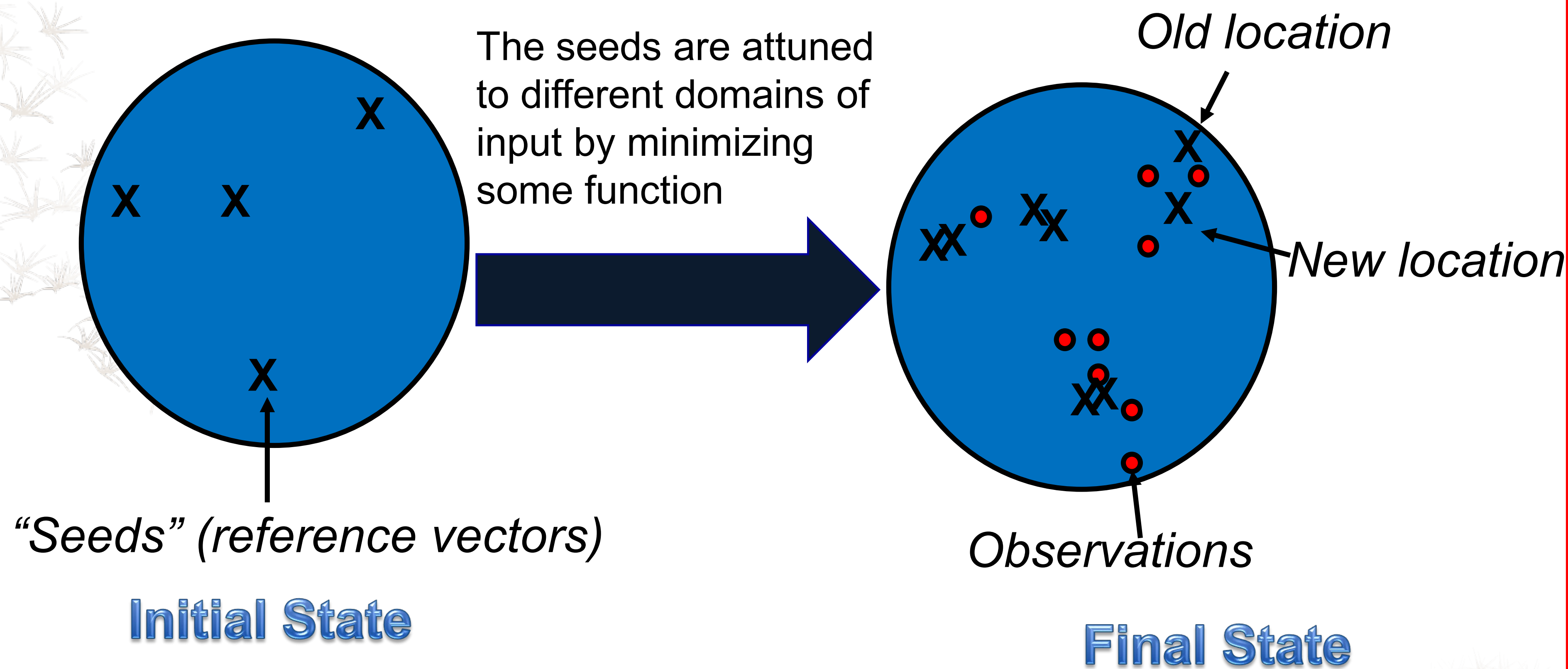
Optimization Clustering

Optimization (Partitive) Clustering

Optimization clustering partitions a data set into groups by minimizing a specified error criterion. This is done through heuristic algorithms.

This method is useful to tackle big datasets with big number of observations.

Optimization (Partitive) Clustering



It does not depend on previously found clusters and scales up linearly with the number of observations.

Optimization Clustering Techniques

- *k*-means clustering (FASTCLUS)

```
PROC FASTCLUS <MAXC= | RADIUS=><options>;
        VAR variables;
RUN;
```

- Nonparametric clustering (MODECLUS)

```
PROC MODECLUS METHOD=method <options>;
        VAR variables;
RUN;
```

Method=0, **1**, …, or 6

- Fuzzy (Q-technique) clustering (FACTOR)

```
PROC FACTOR <options>;
        VAR variables;
RUN;
```

K-means Clustering (FASTCLUS)

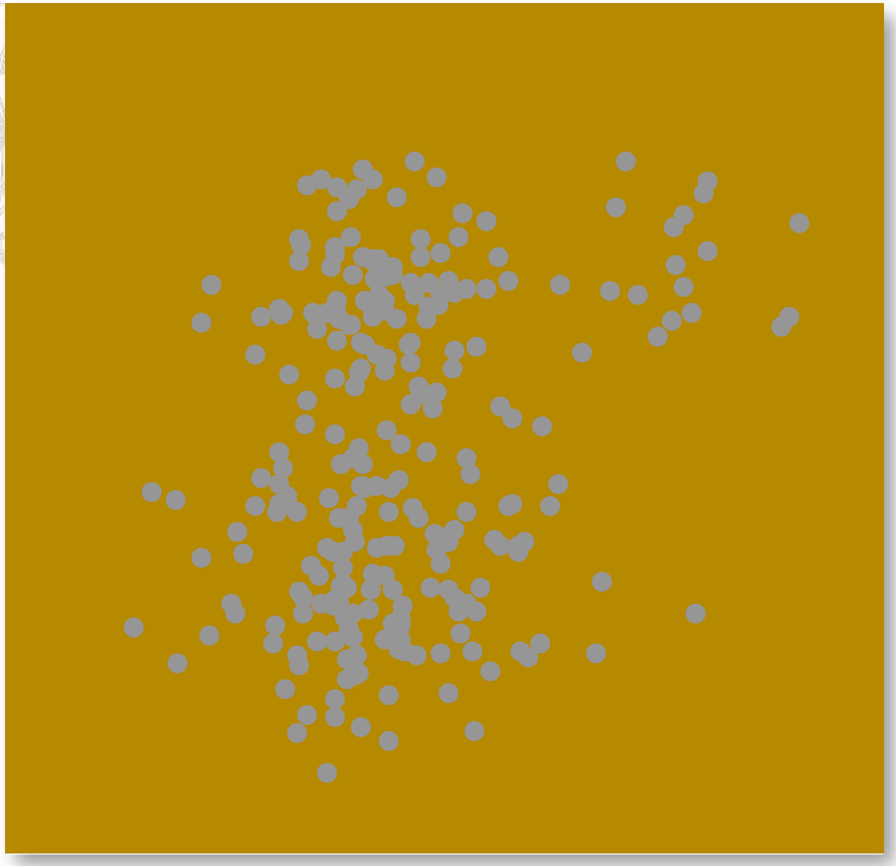
- The best-known optimization clustering algorithm.
- Good for large data sets (> 100 observations).
- Fast as the SAS procedure name (FASTCLUS) implies.

The *K*-means Procedure

1. Select the initial cluster seeds.
2. Each observation is assigned to the nearest seed, forming temporary clusters. The seeds are then replaced by the means of the temporary clusters, and the process is repeated until no significant change occurs in the positions on the cluster means.
3. Each observation is assigned to the nearest seed, forming the final clusters.

k-means procedure

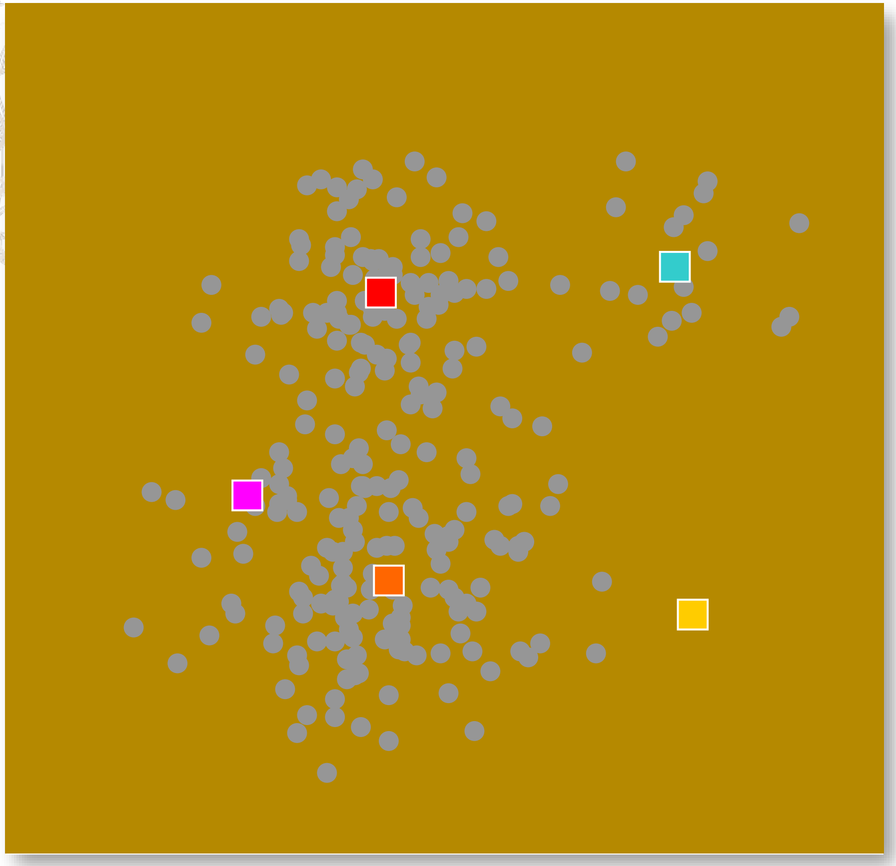
Training Data



1. **Select inputs.**
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. Re-assign cases.
6. Repeat steps 4 and 5 until convergence.

k-means procedure

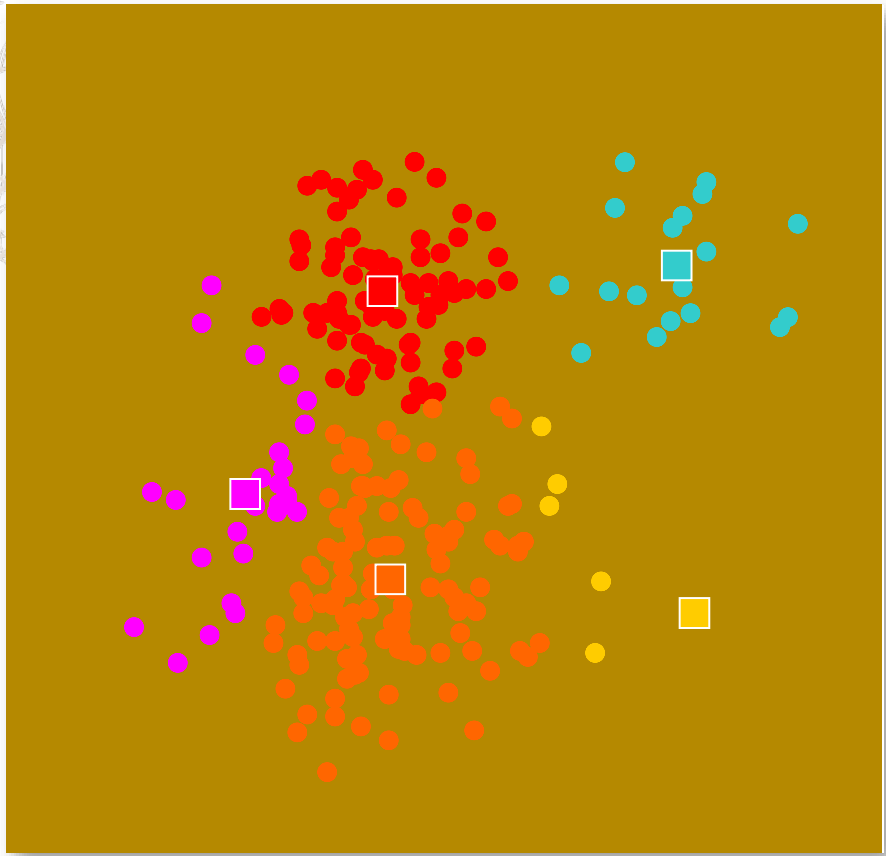
Training Data



1. Select inputs.
2. **Select *k* cluster centers.**
3. Assign cases to closest center.
4. Update cluster centers.
5. Re-assign cases.
6. Repeat steps 4 and 5 until convergence.

k-means procedure

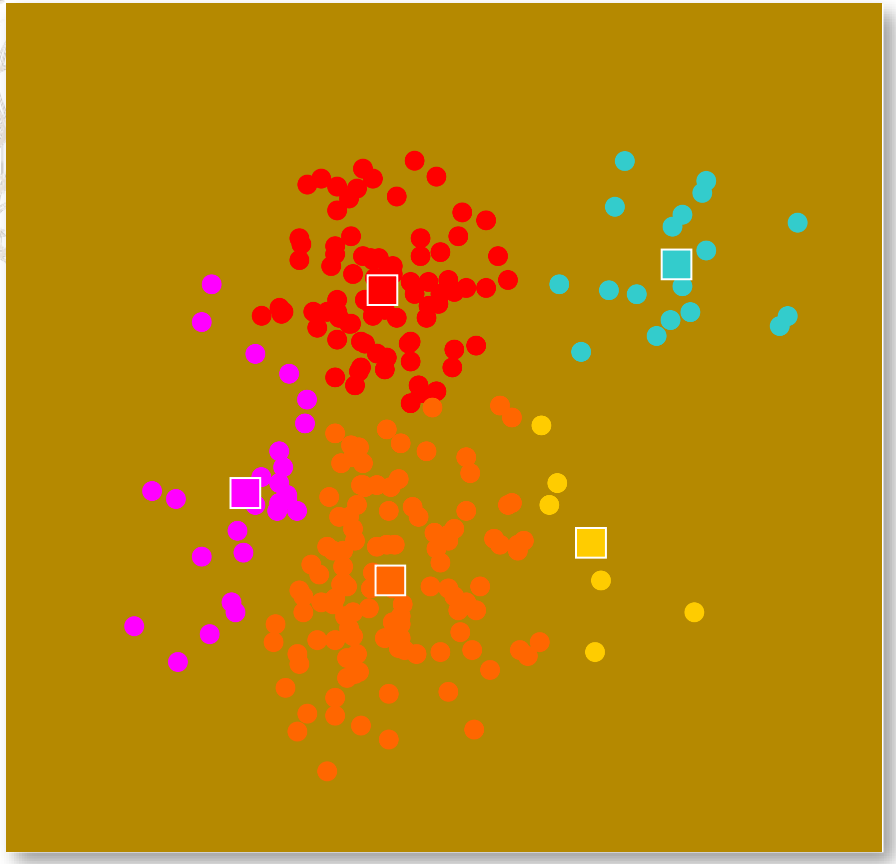
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. **Assign cases to closest center.**
4. Update cluster centers.
5. Re-assign cases.
6. Repeat steps 4 and 5 until convergence.

k-means procedure

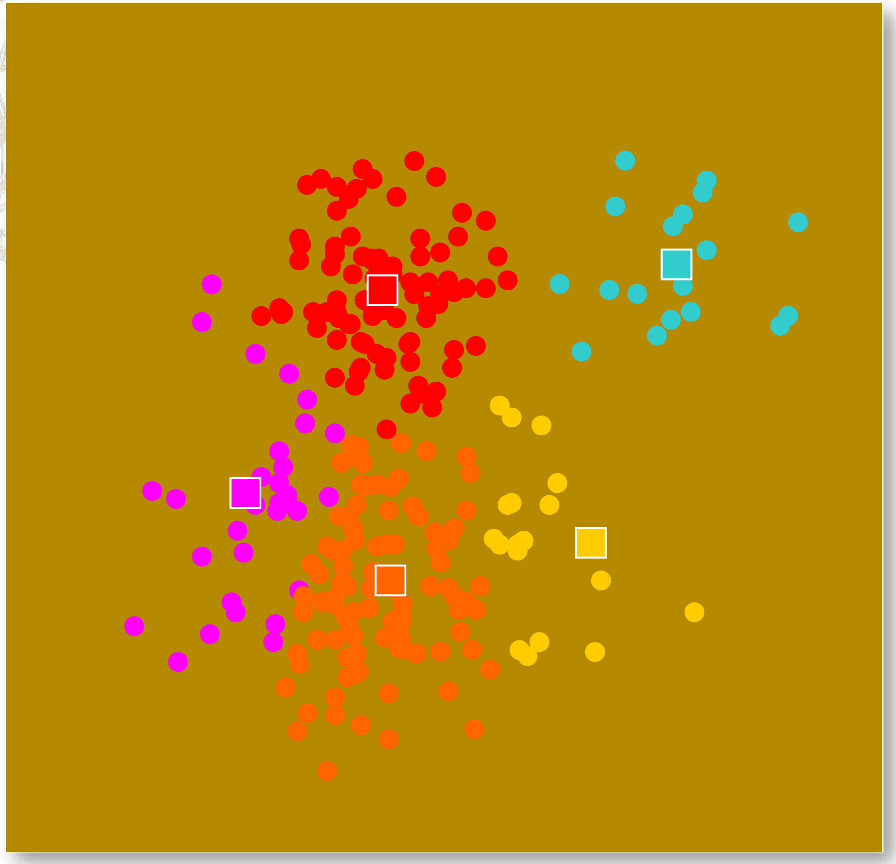
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. **Update cluster centers.**
5. Re-assign cases.
6. Repeat steps 4 and 5 until convergence.

k-means procedure

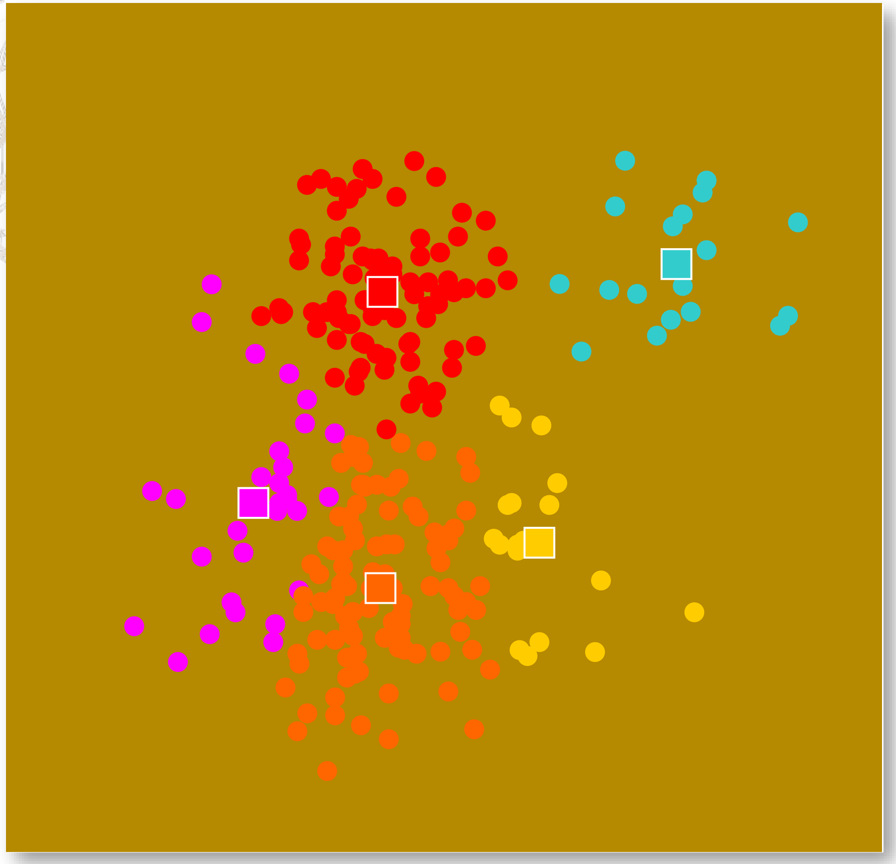
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. **Re-assign cases.**
6. Repeat steps 4 and 5 until convergence.

k-means procedure

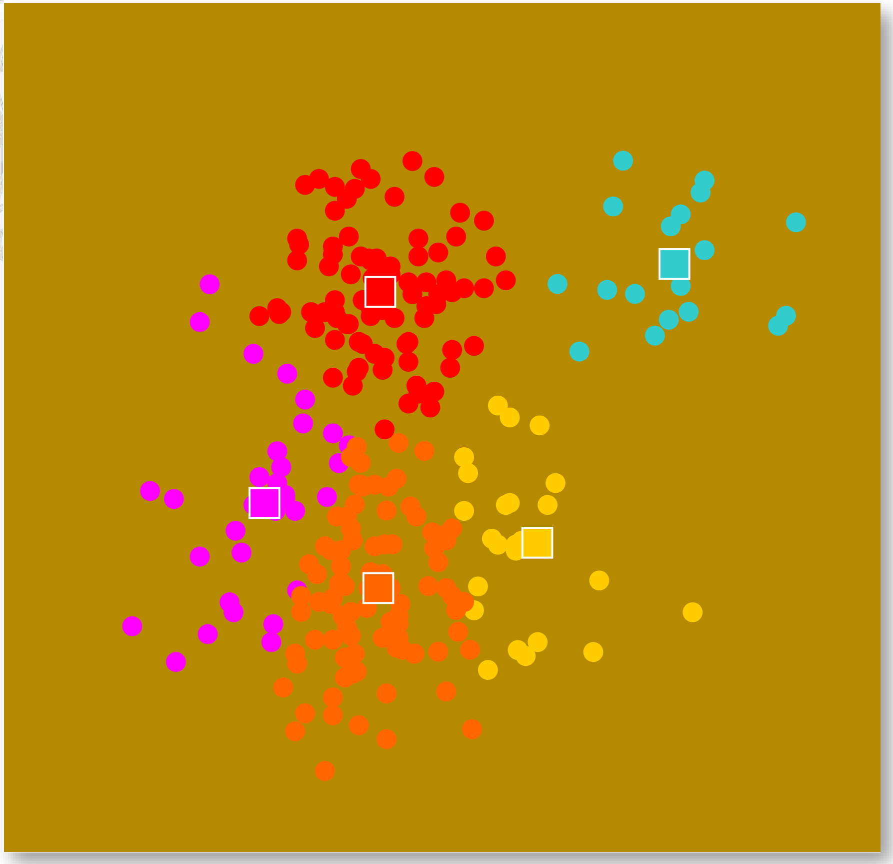
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. **Update cluster centers.**
5. Re-assign cases.
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

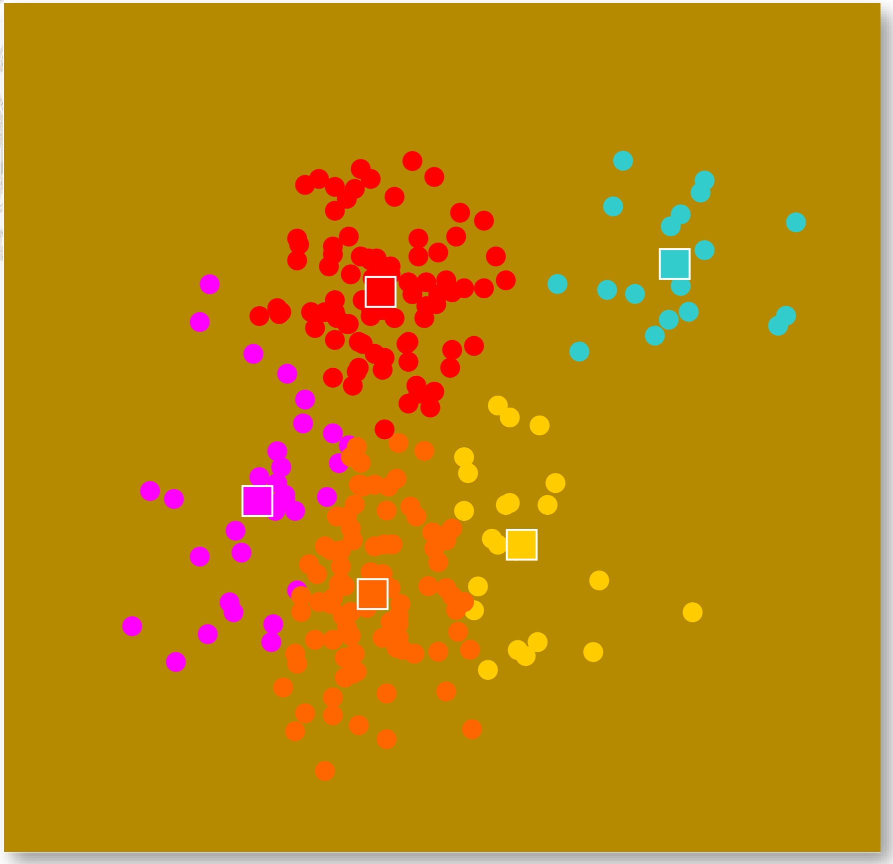
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. **Re-assign cases.**
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

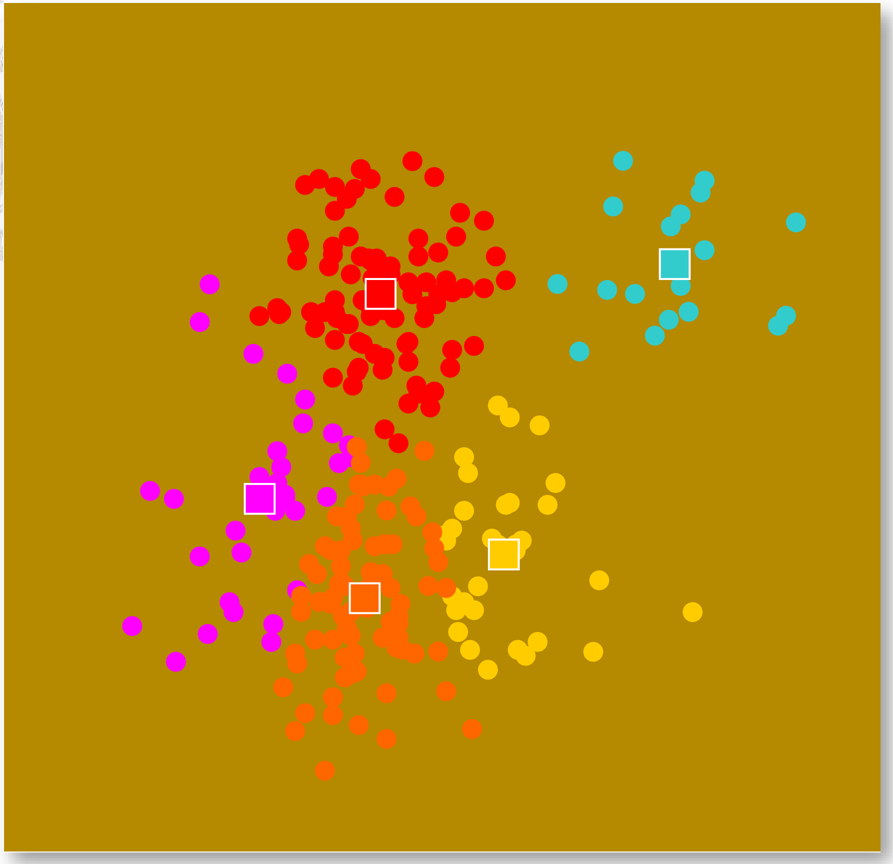
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. **Update cluster centers.**
5. Re-assign cases.
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

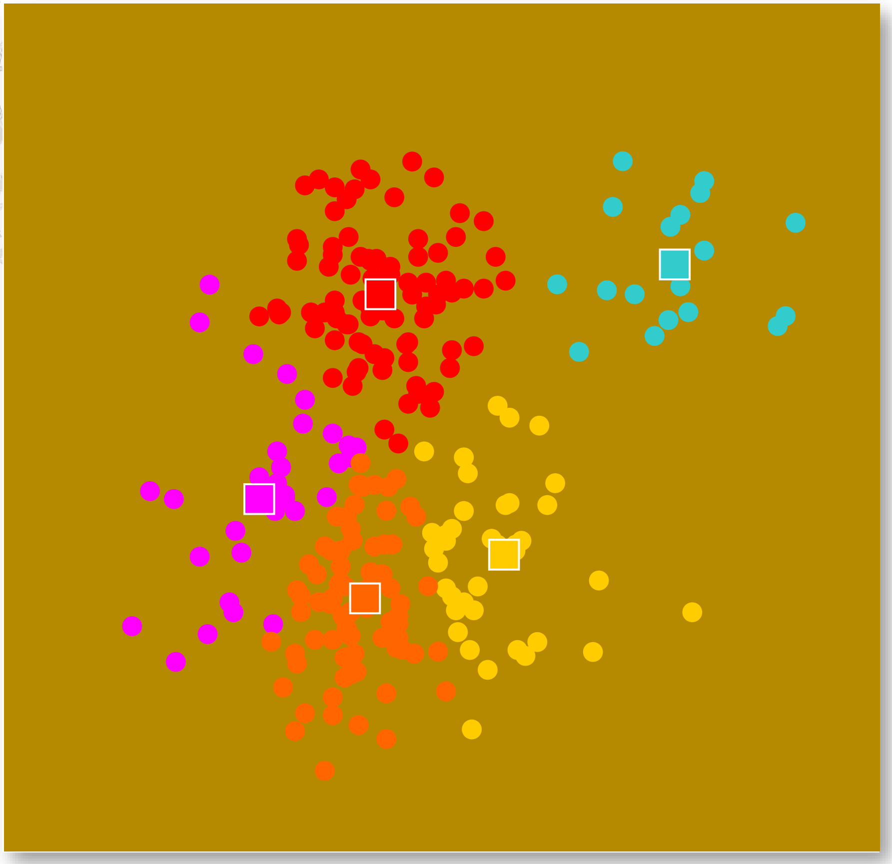
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. **Re-assign cases.**
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

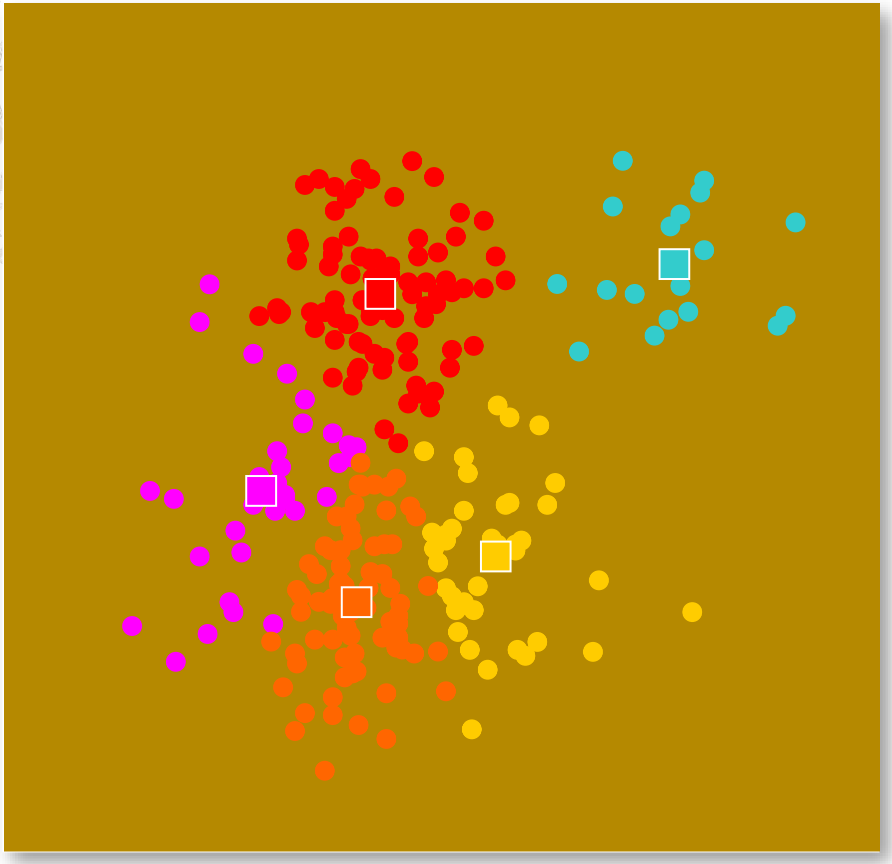
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. **Update cluster centers.**
5. Re-assign cases.
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

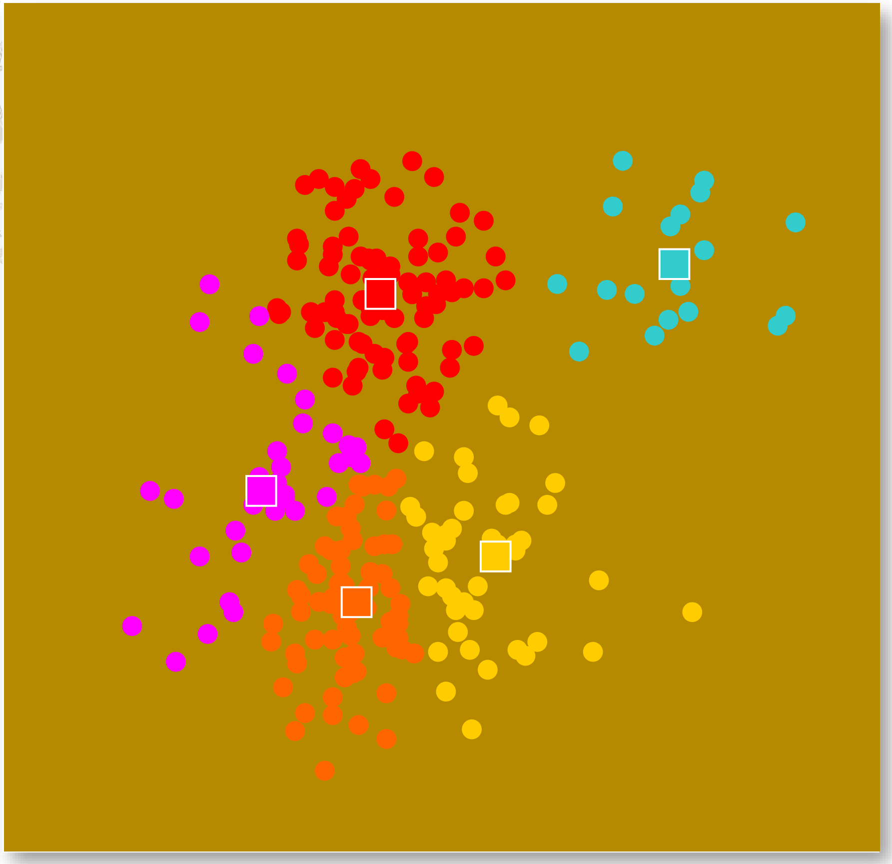
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. **Re-assign cases.**
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

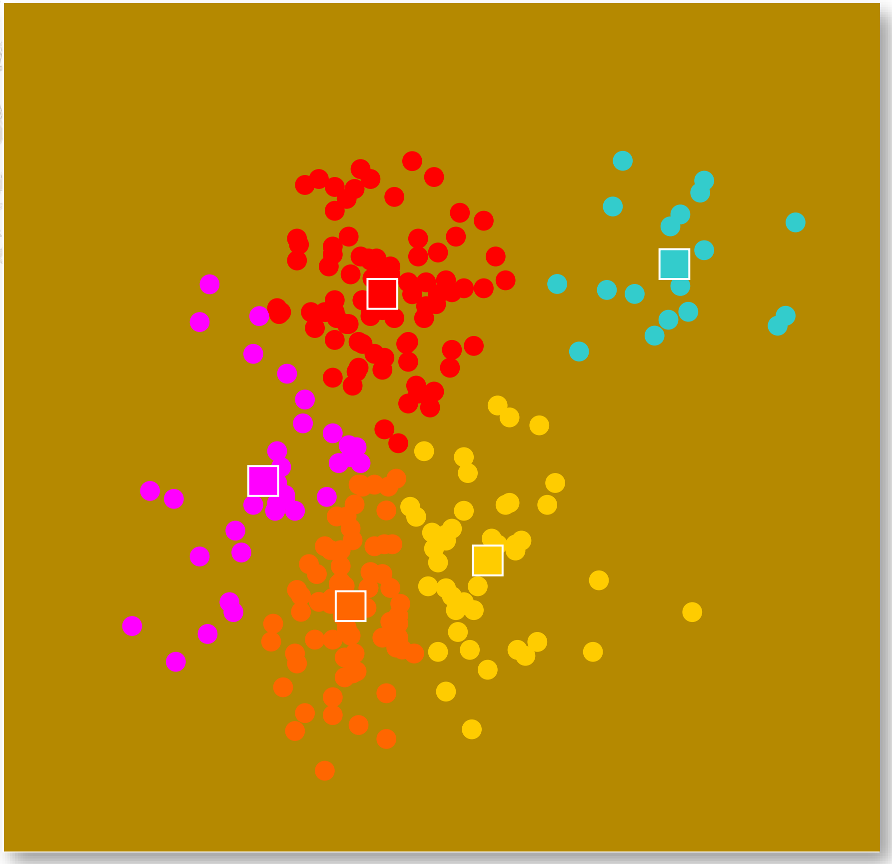
Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. **Re-assign cases.**
6. **Repeat steps 4 and 5 until convergence.**

k-means procedure

Training Data



1. Select inputs.
2. Select *k* cluster centers.
3. Assign cases to closest center.
4. Update cluster centers.
5. Re-assign cases.
6. Repeat steps 4 and 5 until **convergence**.

K-means clustering: FASTCLUS procedure

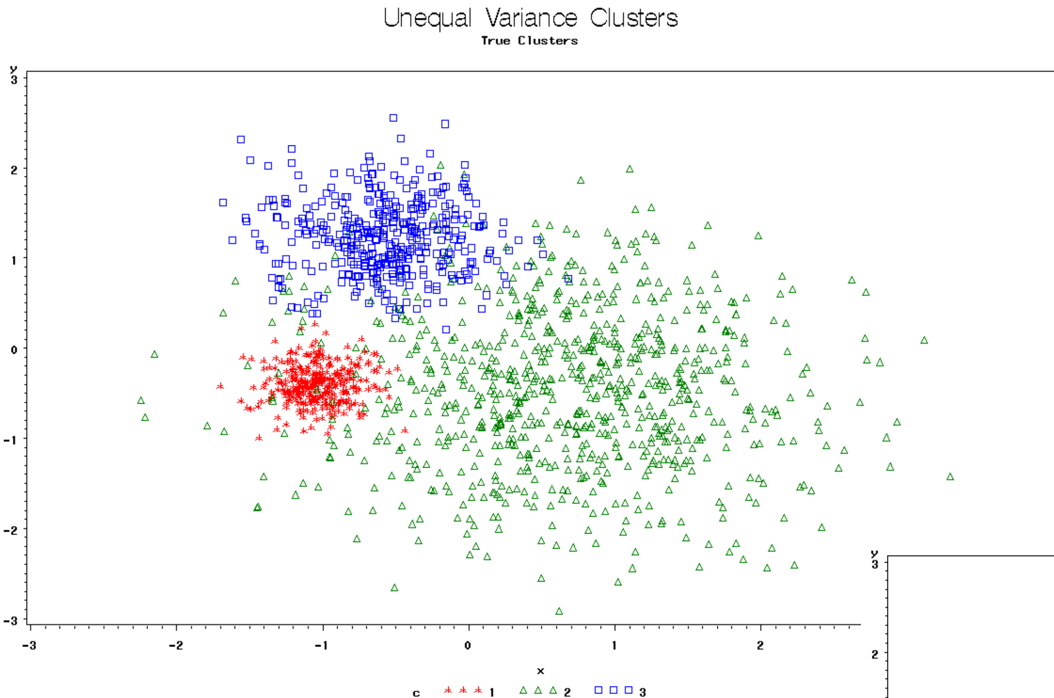
```
PROC FASTCLUS <MAXC= | RADIUS=><options>;
      VAR variables;
RUN;
```

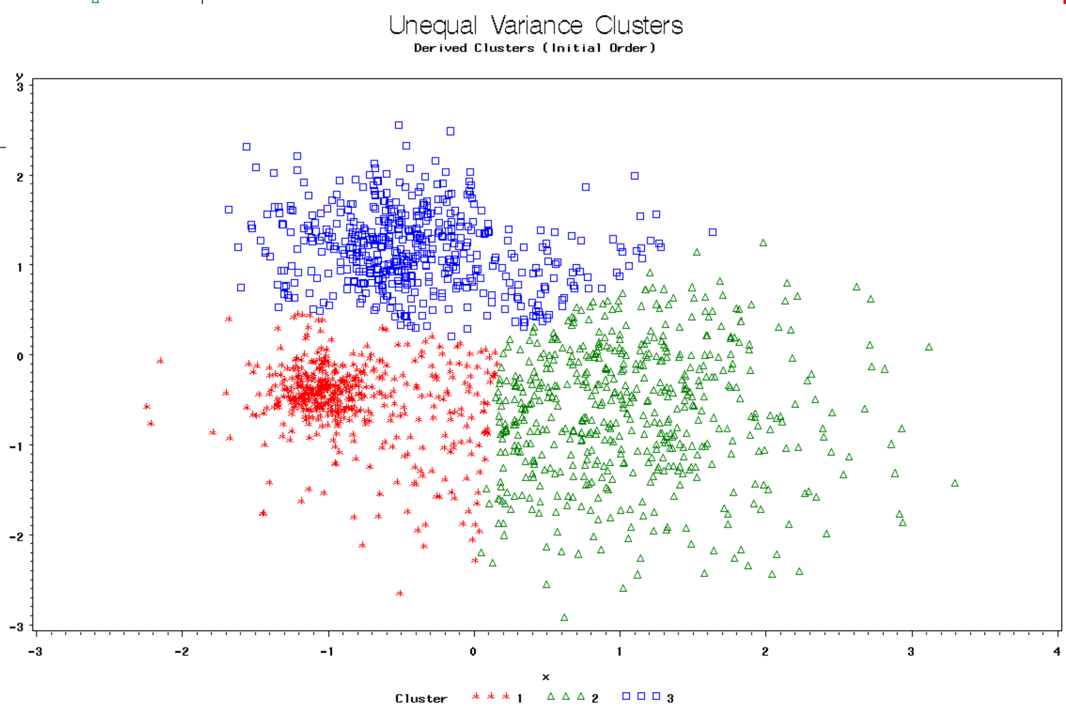
VAR numeric variables to be used
MAXC= maximum number of clusters allowed (default value =100)
RADIUS= no observation is considered as a new seed unless its minimum distance to previous seeds exceeds the value given by the RADIUS= option

K-means Clustering Demo

```
%let inputs=x y;
%let group=c;
proc stdize data=teaching.unequal method=std
         out=unequal;
  var &inputs;
run;
title 'Unequal Variance Clusters';
title2 'True Clusters';
proc gplot data=unequal;
         plot y*x=c;
run;
title2 'K-Means Clustering';
proc fastclus data=unequal maxc=3 radius=1 least=2 out=clusout1;
  var &inputs;
run;
title2 'Derived Clusters';
proc gplot data=clusout1;
         plot y*x=cluster;
run;
```

K-means Clustering: a 3-cluster Example





K-means Clustering Demo

```
%let inputs=x y;
%let group=c;
proc stdize data=teaching.ring method=std        out=unequal;
   var &inputs;
run;
title 'Unequal Variance Clusters';
title2 'True Clusters';
proc gplot data=unequal;
            plot y*x=c;
run;
title2 'K-Means Clustering';
proc fastclus data=unequal maxc=3 radius=1 least=2 out=clusout1;
   var &inputs;
run;
title2 'Derived Clusters';
proc gplot data=clusout1;
            plot y*x=cluster;
run;
```

K-means Clustering: Different maxc= Values