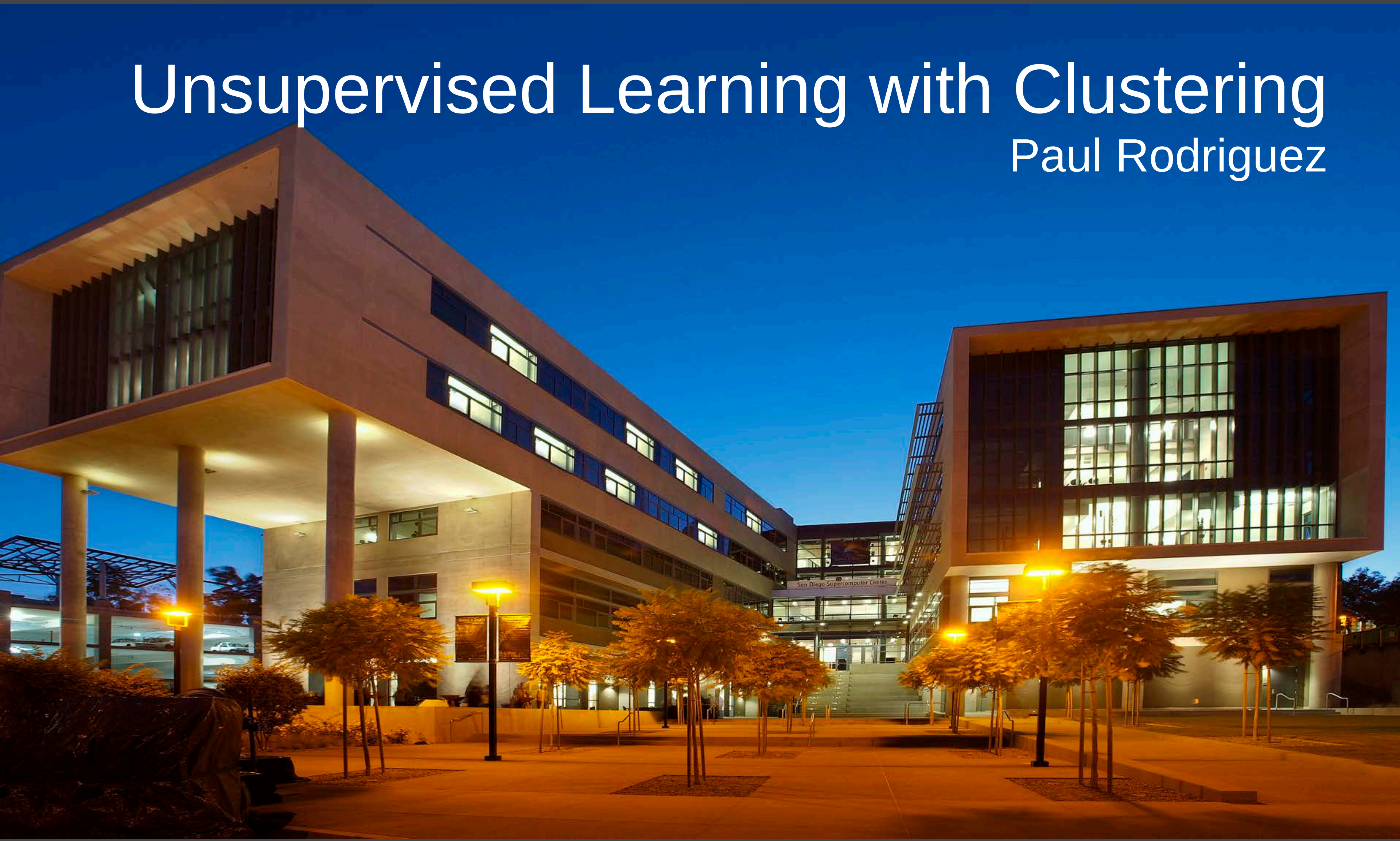
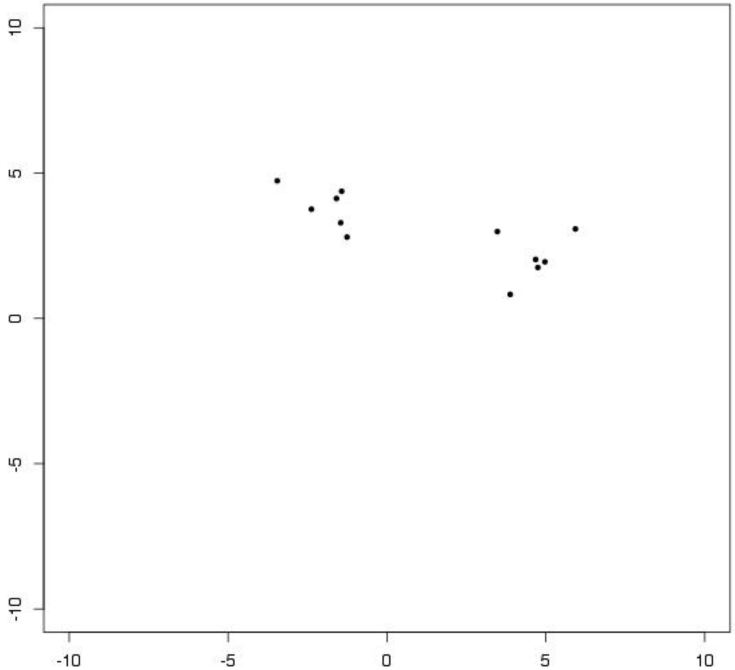
Unsupervised Learning with Clustering

Paul Rodriguez



Clustering Idea

- **Given a set of data can we find a natural grouping?**



X_2

X_1

Essential R commands:

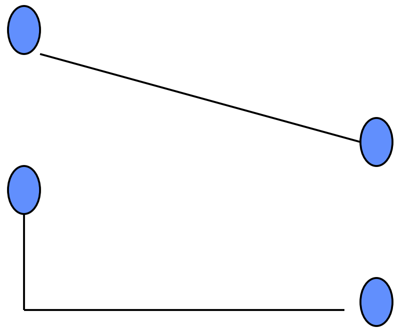
```
D     =rnorm(12,0,1) #generate 12
                     #random normal
X1    =matrix(D,6,2) #put into 6x2 matrix
X1[,1]=X1[,1]+4;     #shift center
X1[,2]=X1[,2]+2;

#repeat for another set of points

#bind data points and plot
plot(rbind(X1,X2),
          xlim=c(-10,10),ylim=c(-10,10));
```

Why Clustering

- **A good grouping implies some structure**
- **In other words, given a good grouping, we can then:**
  - Interpret and label clusters
  - Identify important features
  - Characterize new points by the closest cluster
  - Use the cluster assignments as a summary of the data

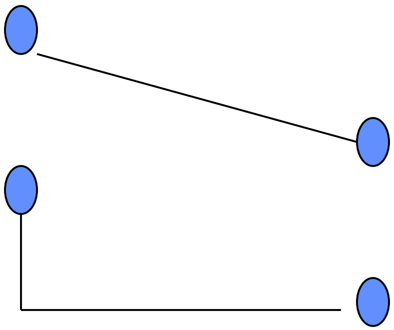
Clustering Objective

- **Objective: find subsets that are similar within cluster and dissimilar between clusters**
- **Similarity defined by distance measures**
  - Euclidean distance
  - Manhattan distance



Clustering Objective

- **Objective: find subsets that are similar within cluster and dissimilar between clusters**
- **Similarity defined by distance measures**



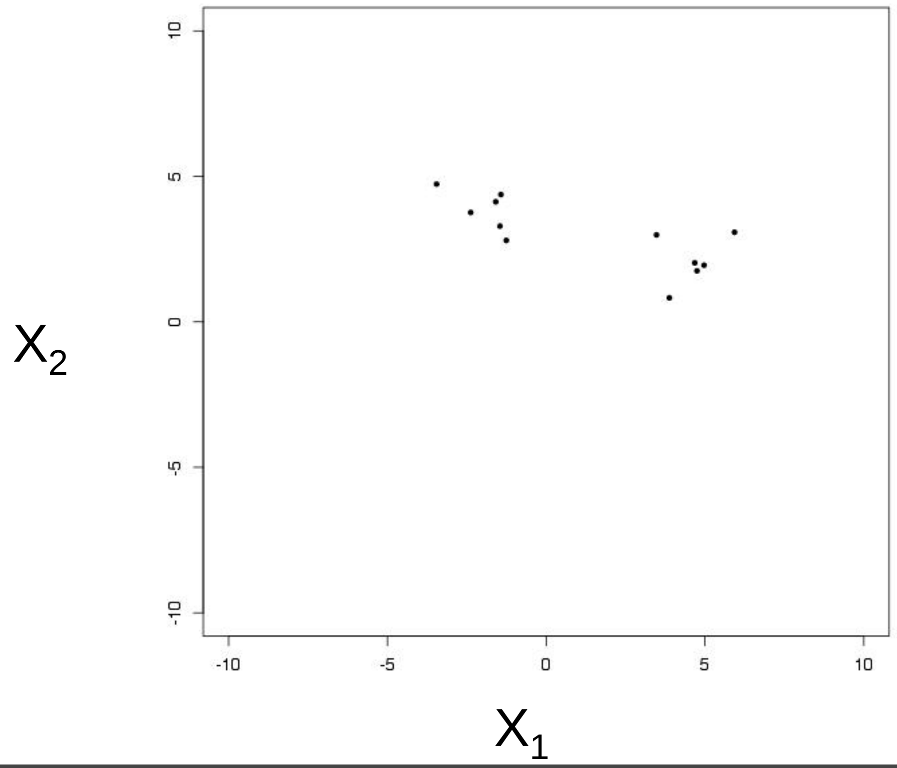
  - Euclidean distance =
    sqrt[(a1 - b1)^2+ (a2 – b2)^2+...)]
  - Manhattan distance
    [|a1 - b1|+ |a2 – b2|+...)]

Kmeans Clustering

- **A simple, effective, and standard method**
  - Start with K initial cluster centers
  - Loop:
    - Assign each data point to nearest cluster center
    - Calculate mean of cluster for new center
  - Stop when assignments don’t change
- **Issues:**
  - How to choose K?
  - How to choose initial centers?
  - Will it always stop?

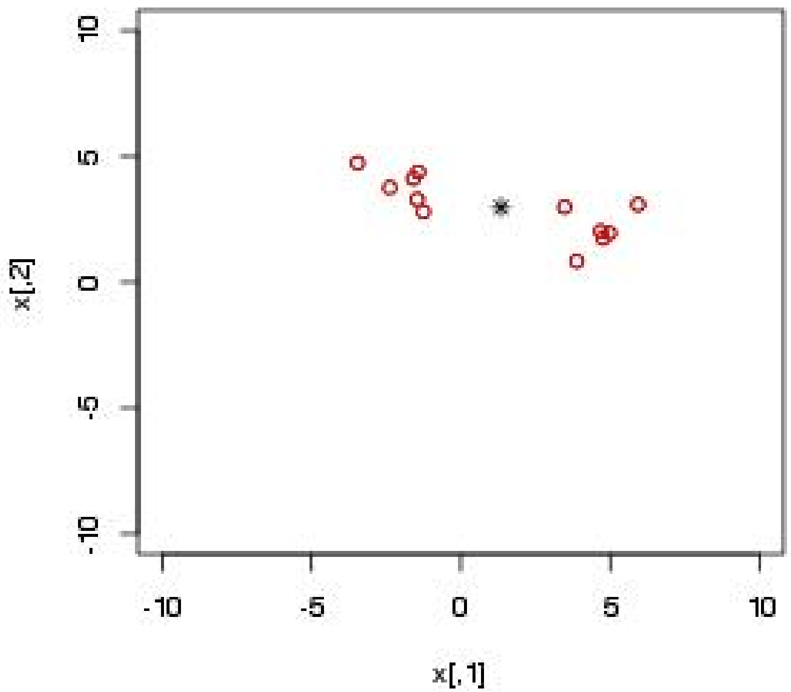
Kmeans Example

- **For K=1, using Euclidean distance, where will the cluster center be?**



Kmeans Example

For K=1, the overall mean minimizes Sum Squared Error (SSE), aka Euclidean dist. squared

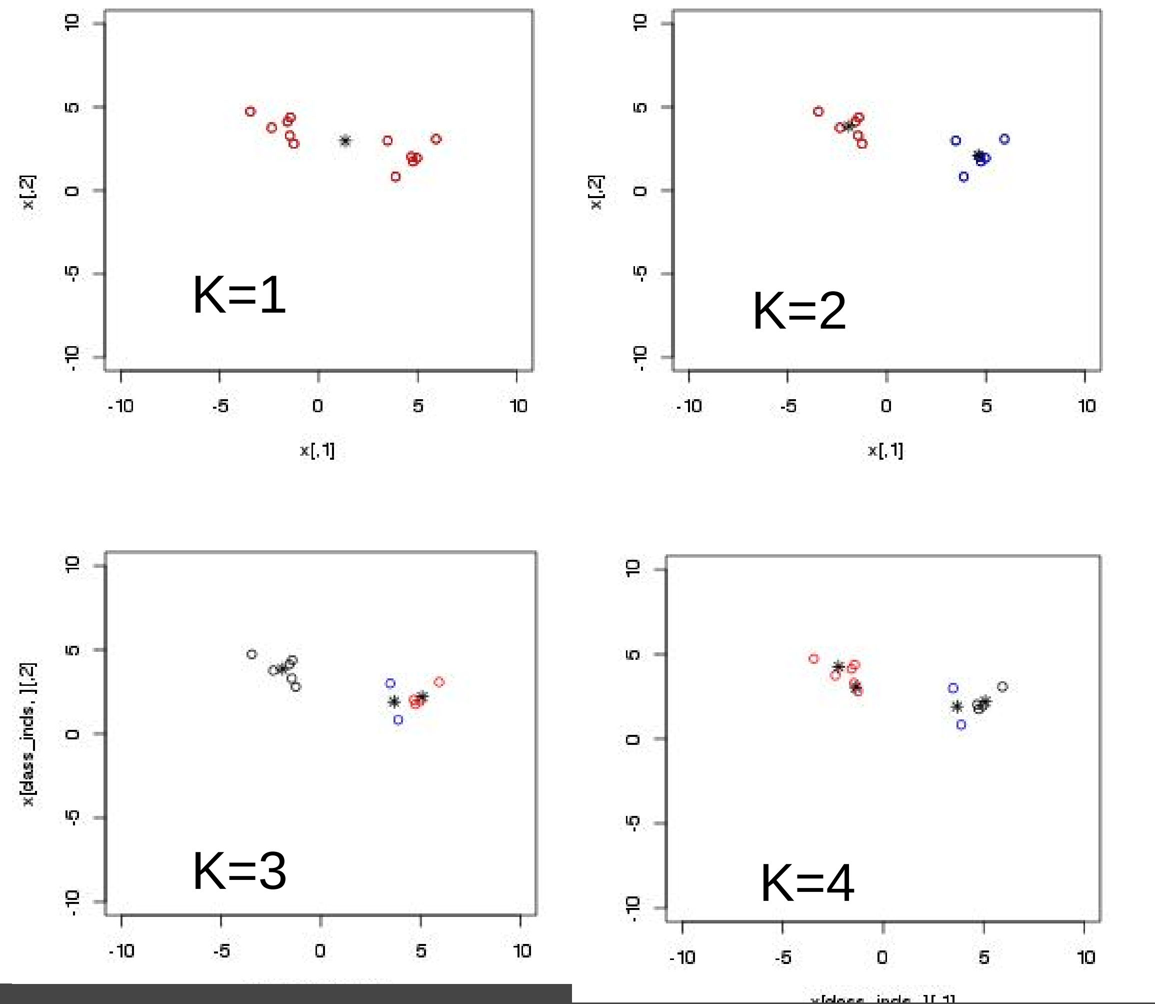


Essential R commands:
Kresult = kmeans(X,1,10,1)

#choose 1 data point as initial K centers
#10 is max loop iterations
#1 is number of initial sets to try

#Kresult is an R object with subfields
Kresult$cluster   #cluster assignments
Kresults$tot.withinss # tot within SSE

Kmeans Example

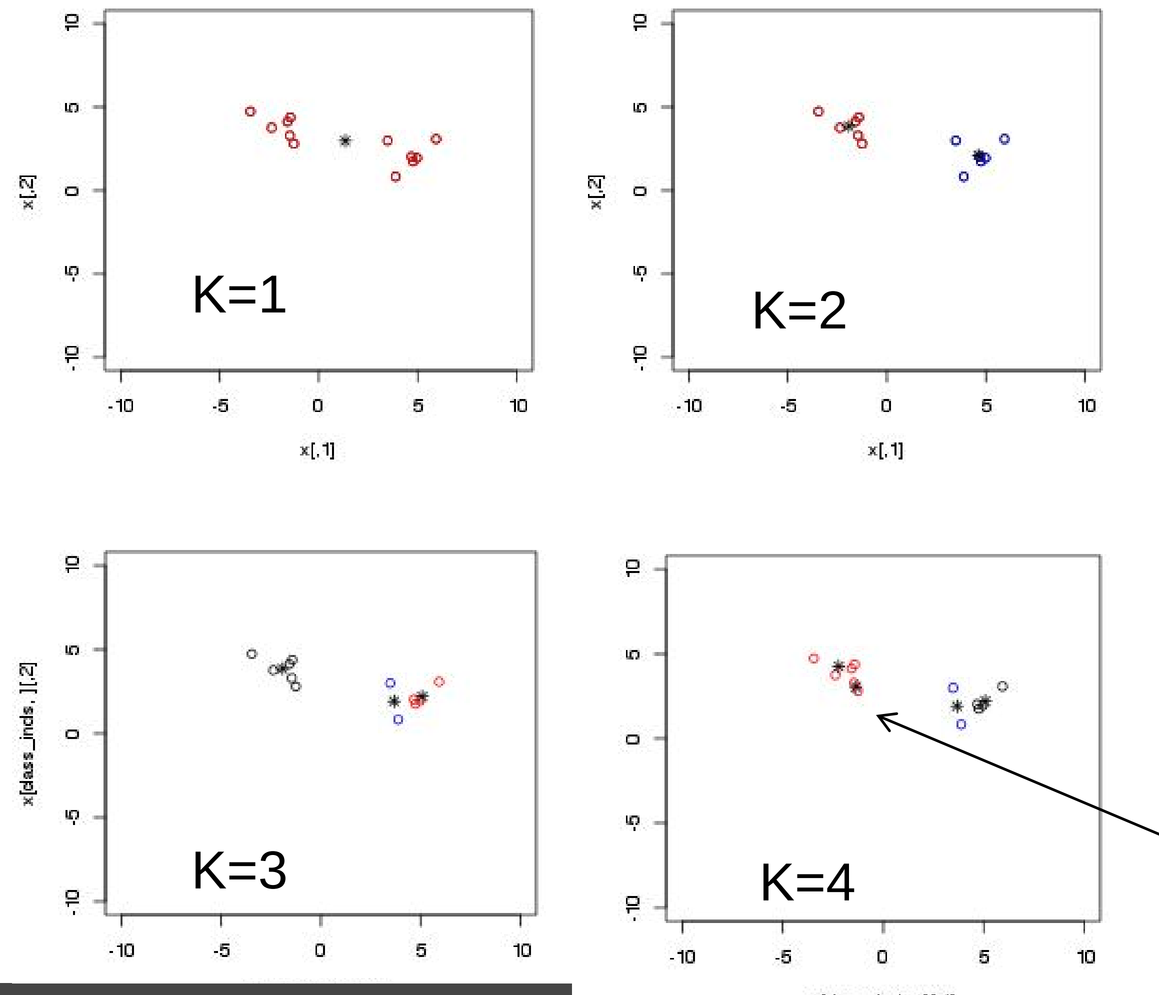


Essential R commands:
inds=which(Kresult$cluster==K)

plot(X[inds,],col2use=“red”);

…

Kmeans Example



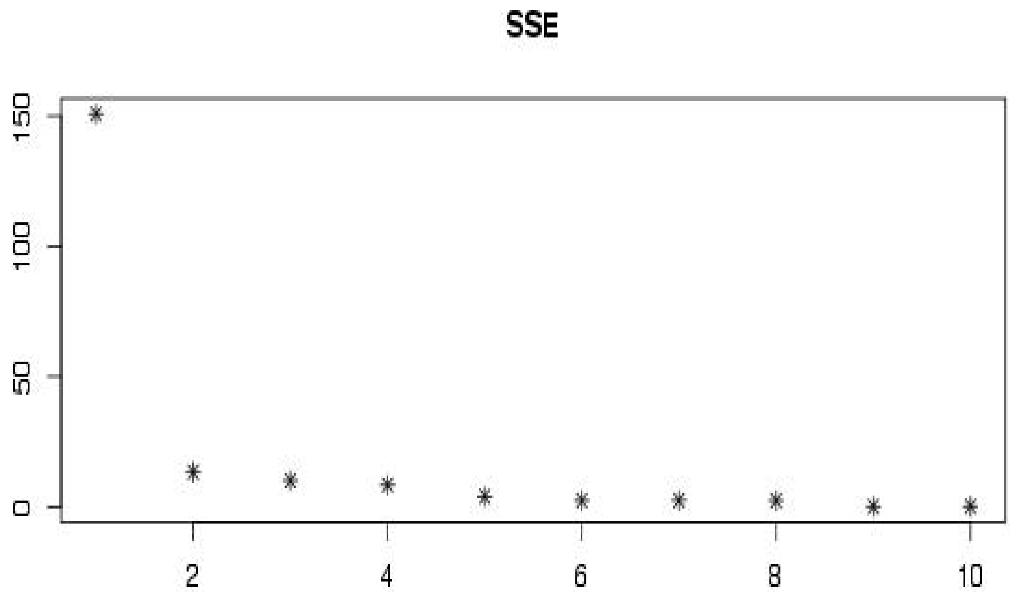
Essential R commands:
inds=which(Kresult$cluster==K)

plot(X[inds,],col2use=“red”);

…

As K increases individual points get a cluster

Choosing K for Kmeans



Total Within Cluster SSE

K=1 to 10

Essential R commands:

for (num_k in 1:10) {
Kres=kmeans(X,num_k,10,1);

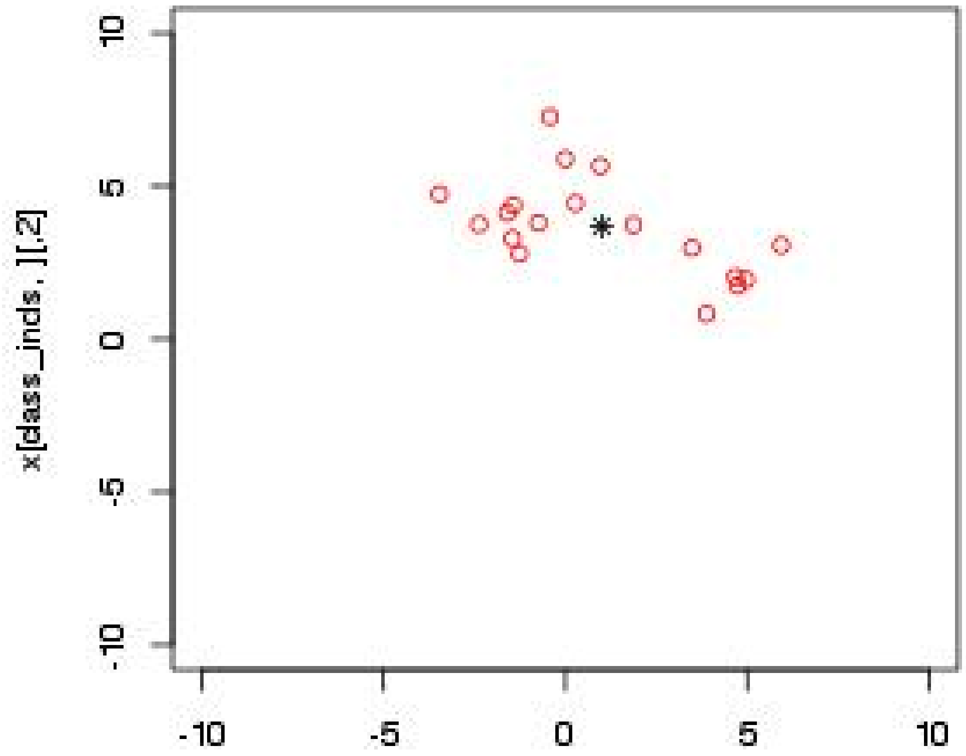
Save and then plot
Kres$tot.withinss

…

- Not much improvement after K=2 (“elbow”)

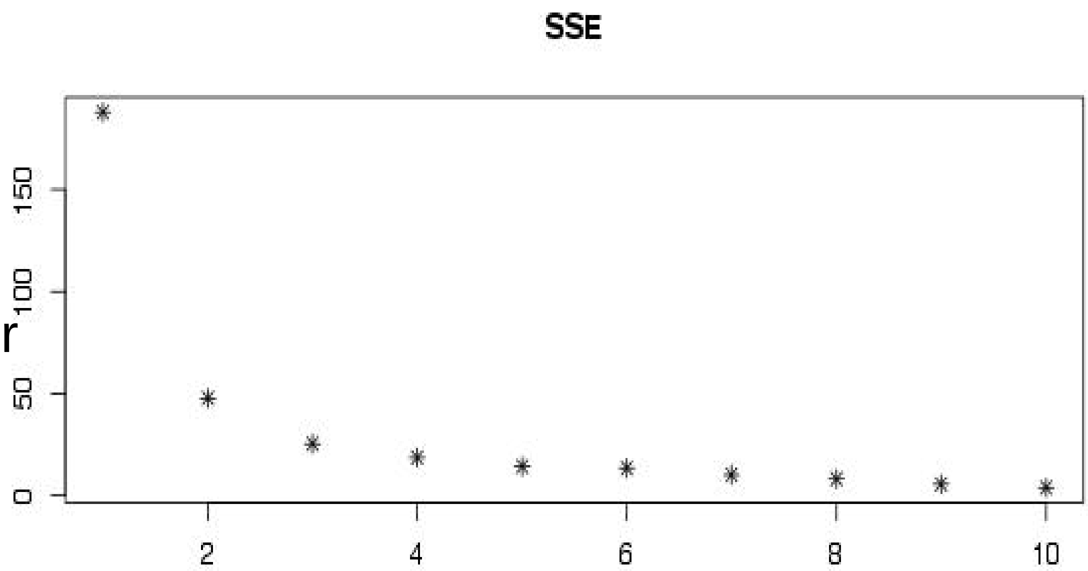
Kmeans Example – more points

How many clusters should there be?



Choosing K for Kmeans

Total Within Cluster SSE



K=1 to 10

- Smooth decrease at K $\geq$ 2, harder to choose
- In general, smoother decrease => less structure

Kmeans Guidelines

- **Choosing K:**

  "Elbow" in total-within-cluster SSE as K=1…N

  Cross-validation: hold out points, compare fit as K=1…N

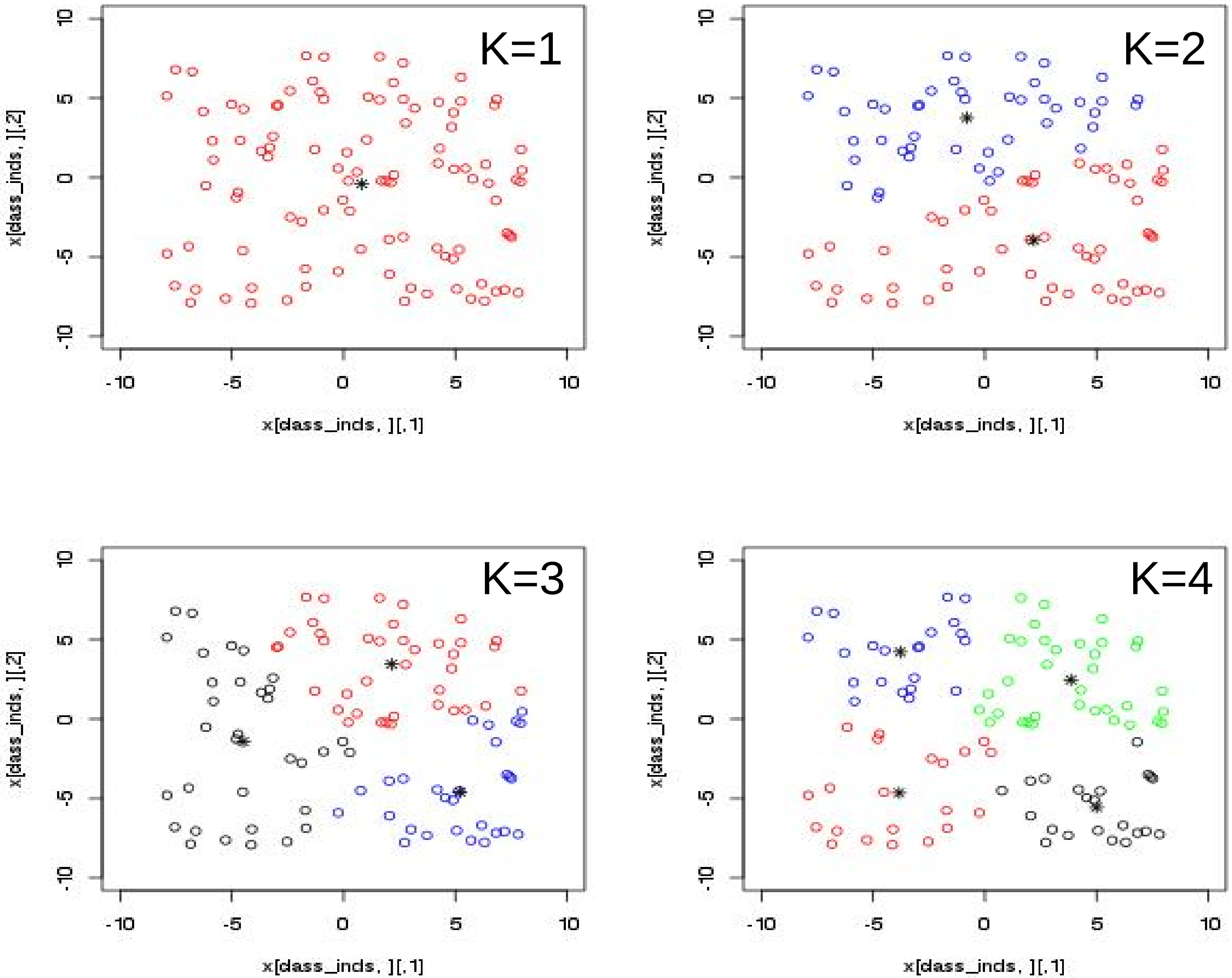
- **Initial starting points and convergence:**

  Kmeans++ algorithm selects good initial clusters that also helps convergence

  10 iterations often good

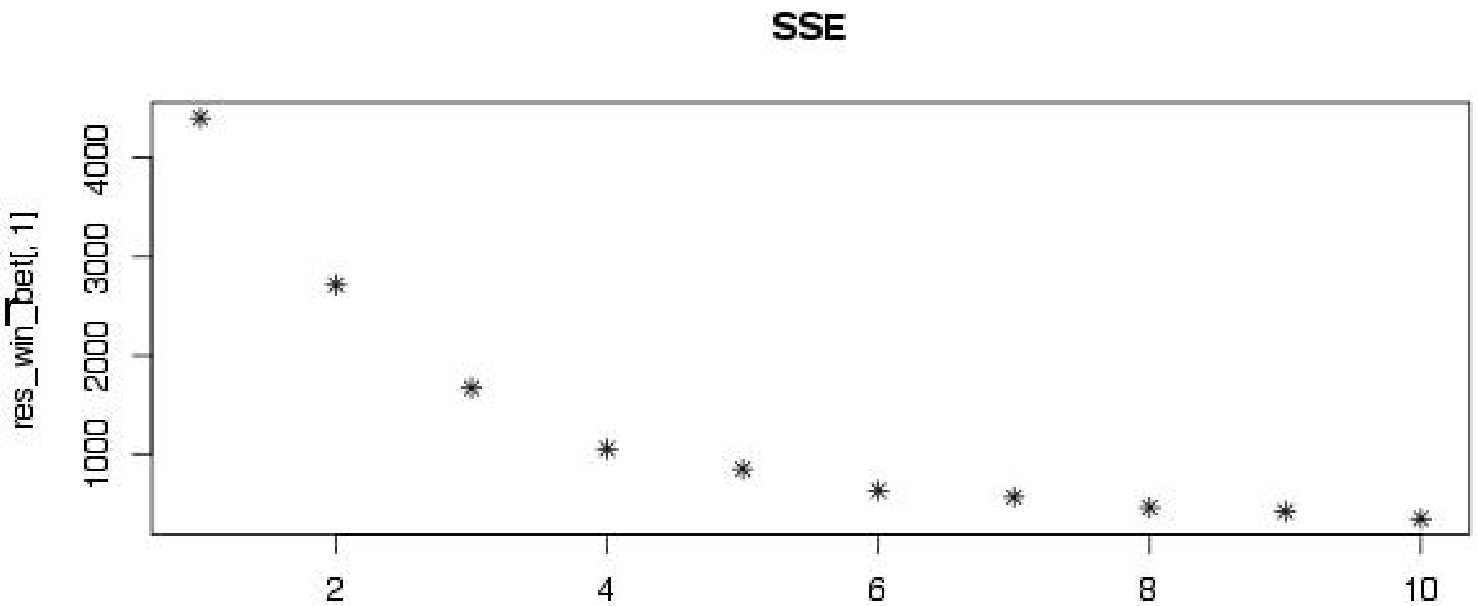
  Can also run several times and choose best result

Kmeans Example: uniform dist.



Choosing K - uniform

Total
Within
Cluster
SSE



K=1 to 10

- Smooth decrease across K => less structure

Kmeans Clustering Issues

- **Scale:**
  - Dimensions with large numbers may dominate distance metrics (so can be good to normalize or scale data)

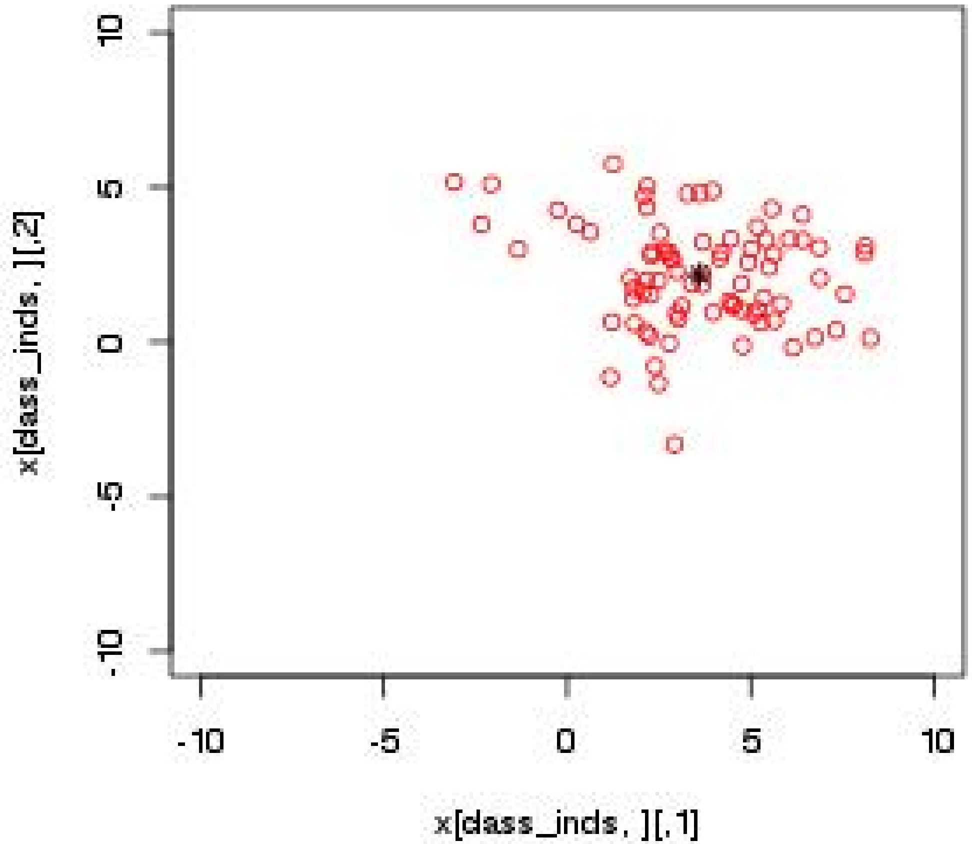
- **Outliers:**
  - Outliers can pull cluster mean

    (K-mediods uses median instead of mean)

Soft Clustering Methods

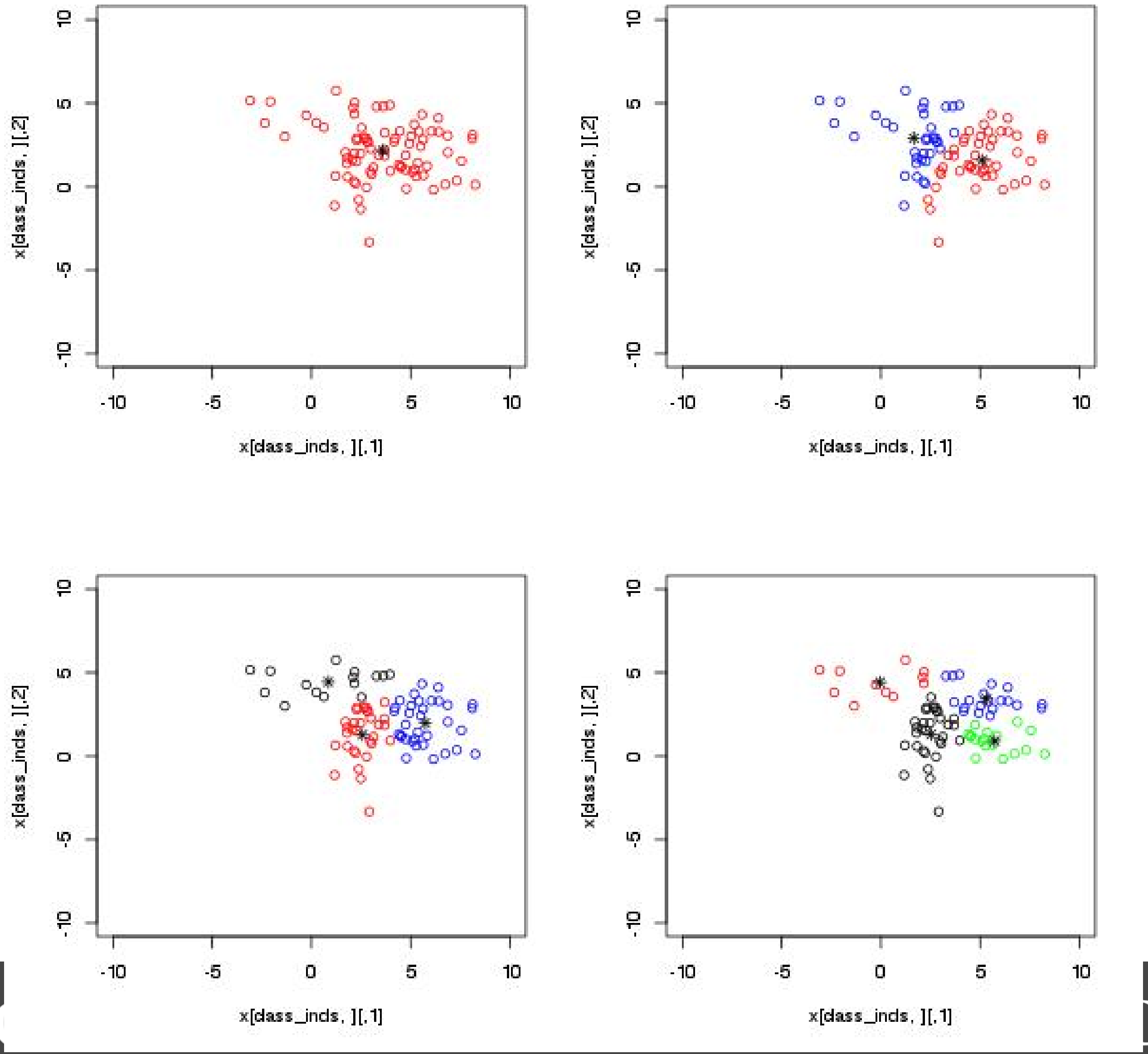
- **Fuzzy Clustering**
  - Kmeans with weighted assignments to all clusters
  - Weights depend on relative distance
  - Find min weighted SSE
- **Expectation-Maximization:**
  - Initialize a mixture of multivariate Gaussian distributions
  - Find means, variances, and mixture weights that maximize probability of data

Kmeans with unequal cluster variance and/or size

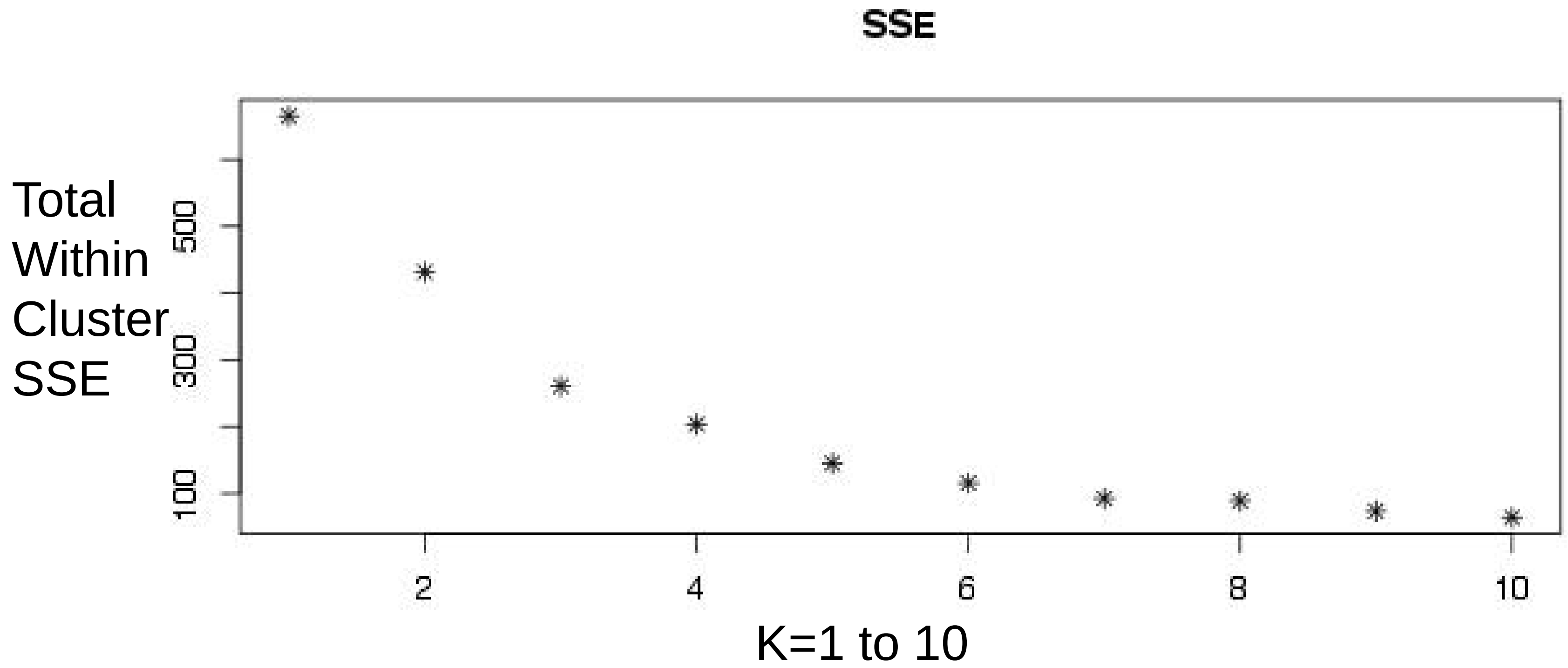


Can you guess K?

Kmeans – unequal cluster variance

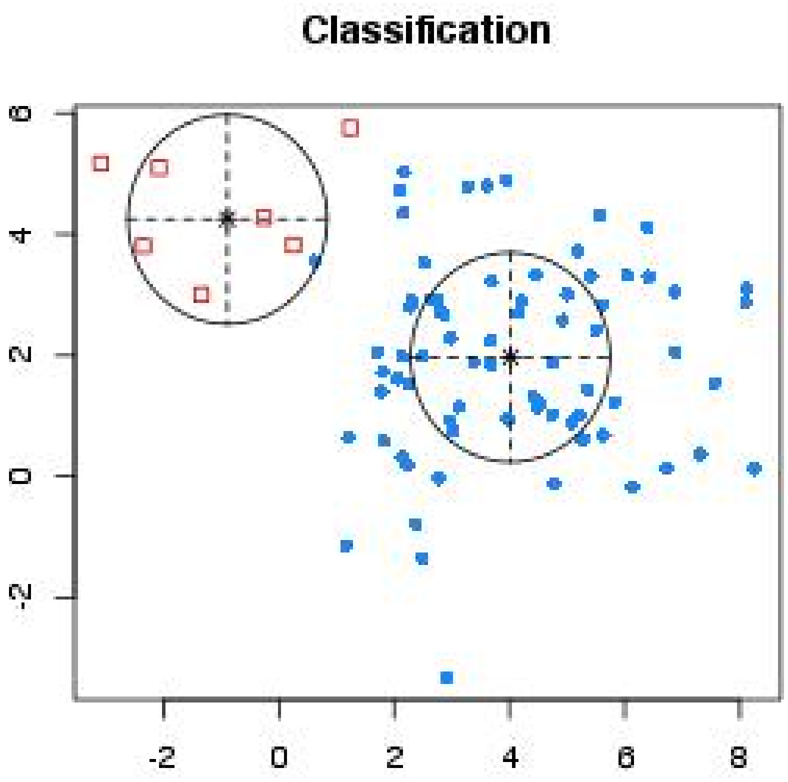


Choosing K – unequal distributions



- Smooth decrease across K => less structure

EM clustering



- Selects K=2
  (either by Information Criterion=
  min of SSE+ K*logN,
  Or by cross-validation)

- Handles unequal variance and/or size

R:
library('mclust')
em_fit=Mclust(x);
plot(em_fit);

Kmeans computations

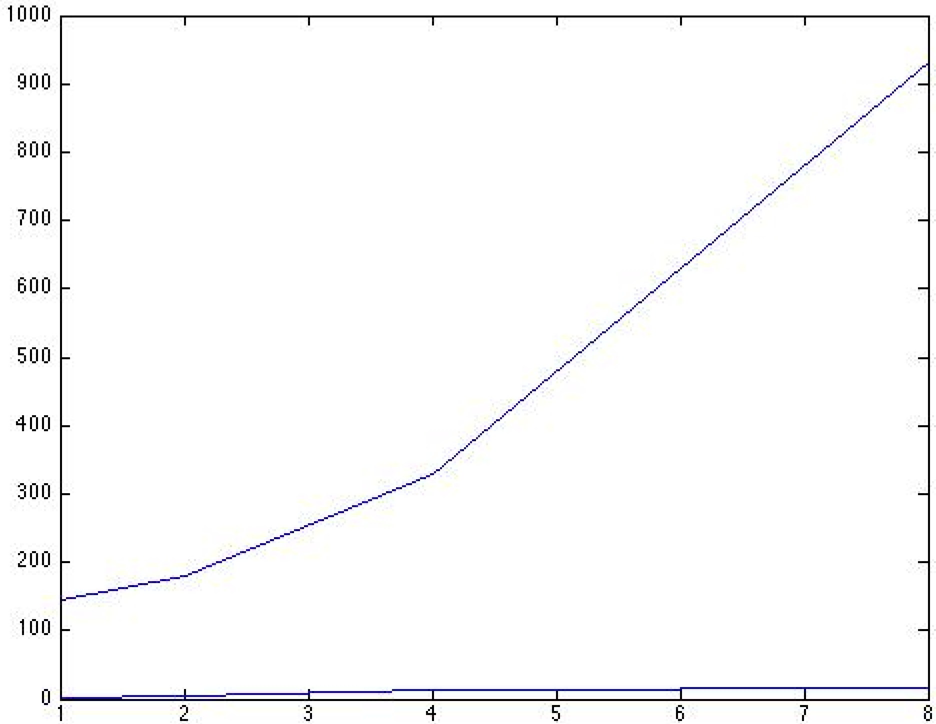
- **Distance of each point to each cluster center**
  - For *N* points, *D* dimensions: each loop requires *N*D*K* operations
- **Update Cluster centers**
  - only track points that change, get change in cluster center
- **But for EM errors to each cluster center update a probability function**

Kmeans vs EM performance

1 Gordon compute node, normal random matrices
R: system.time(Mclust())

15 min

Wall
Time
(secs) 10 min



EM: 1000 pts

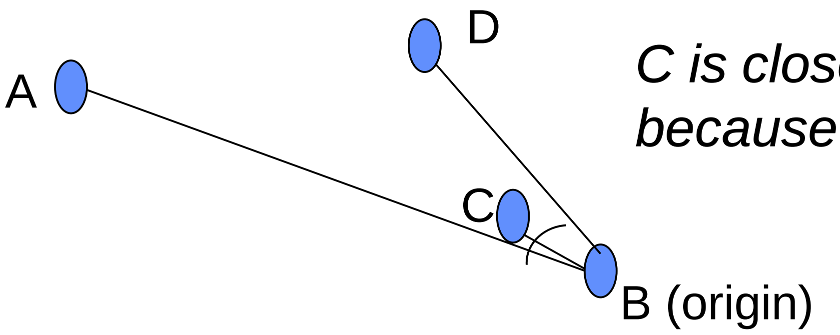
Kmeans:1000 pts

1K 4K 8K

Number of Dimensions (i.e. columns in data matrix)

Other distance measures

- Cosine: take angle difference (good for sparse vectors)



C is closer to A
because $ABC^{\circ} < ABD^{\circ}$

- Mahalanobis: dimensions rescaled by variance

- Jaccard (over sets A,B): $1- (|A \cap B| / |A \cup B|)$
  (e.g. sets of words in documents)

Other distance measures

- Hamming distance: count 1 if values different

  e.g. appropriate for binary strings

010100110
000100010

Total difference is 2

Kmeans big data example

- **45,000 NYTimes articles, 102,000 unique words**

    (UCI Machine Learning repository)

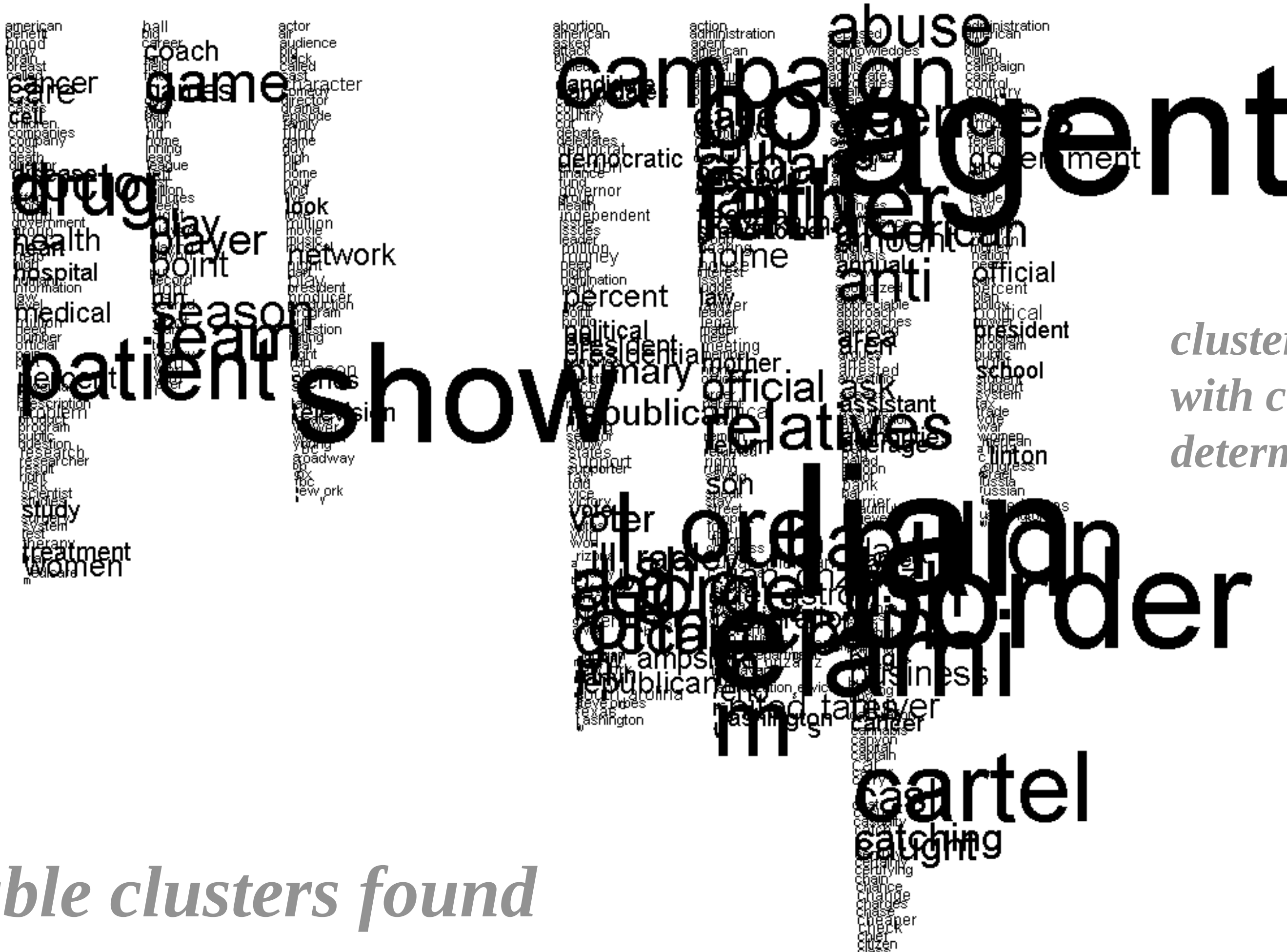
- **Full Data Matrix: 45Kx102K ~ 40Gb**

a and … … … zebra zoo

article 1 0 0 1 0 0 0 2 0 0 … 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
article 2 0 0 0 0 2 0 0 0 0 … 0 0 0 0 3 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
article 3 0 0 0 0 0 0 1 0 0 … 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
…
…
…
…
…
article 45K 0 0 0 0 0 0 0 4 0 0 0 1 0 0 0 1 0 0 0 2 0 0 1 0 0 0 2 0 0 … 0 0 1 0 0

Cell i,j is count of i^{th}-word in j^{th}-article

Kmeans results



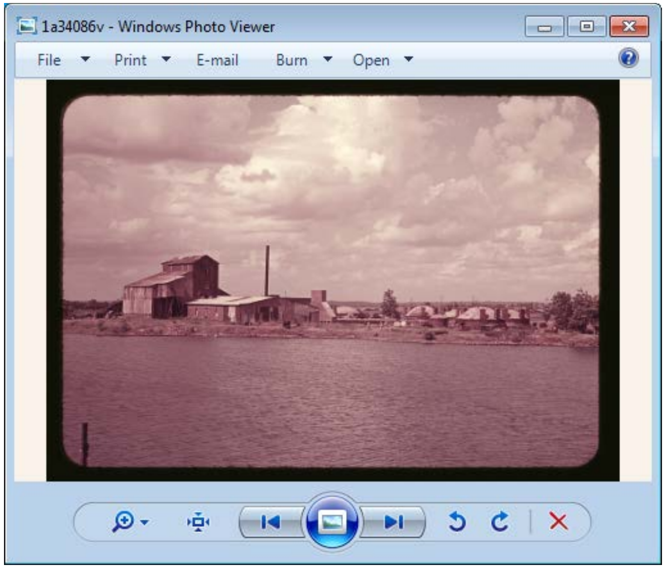
cluster means shown with coordinates determining fontsize

7 viable clusters found

Kmeans for image segmentation

R snippet
get packages
read 1024X718X3 RGB image
convert to matrix 1024*718 X 3
Choose K by trial and error
run Kmeans and display

```
install.packages('ripa')
library('ripa')

source("http://bioconductor.org/biocLite.R")
biocLite()
biocLite("EBImage")

library('EBImage')
im=readImage('1a34086v.jpg')

library('ripa')
img=rgb2grey(im, coefs=c(0.30, 0.59, 0.11))

imgx1  =as.vector(img)
numk=8
km_imx1=kmeans(imgx1,numk,50,1);
img_km_mat =matrix(km_imx1$cluster,dim(im)[1],dim(im)[2])

display(img_km_mat/numk)
```

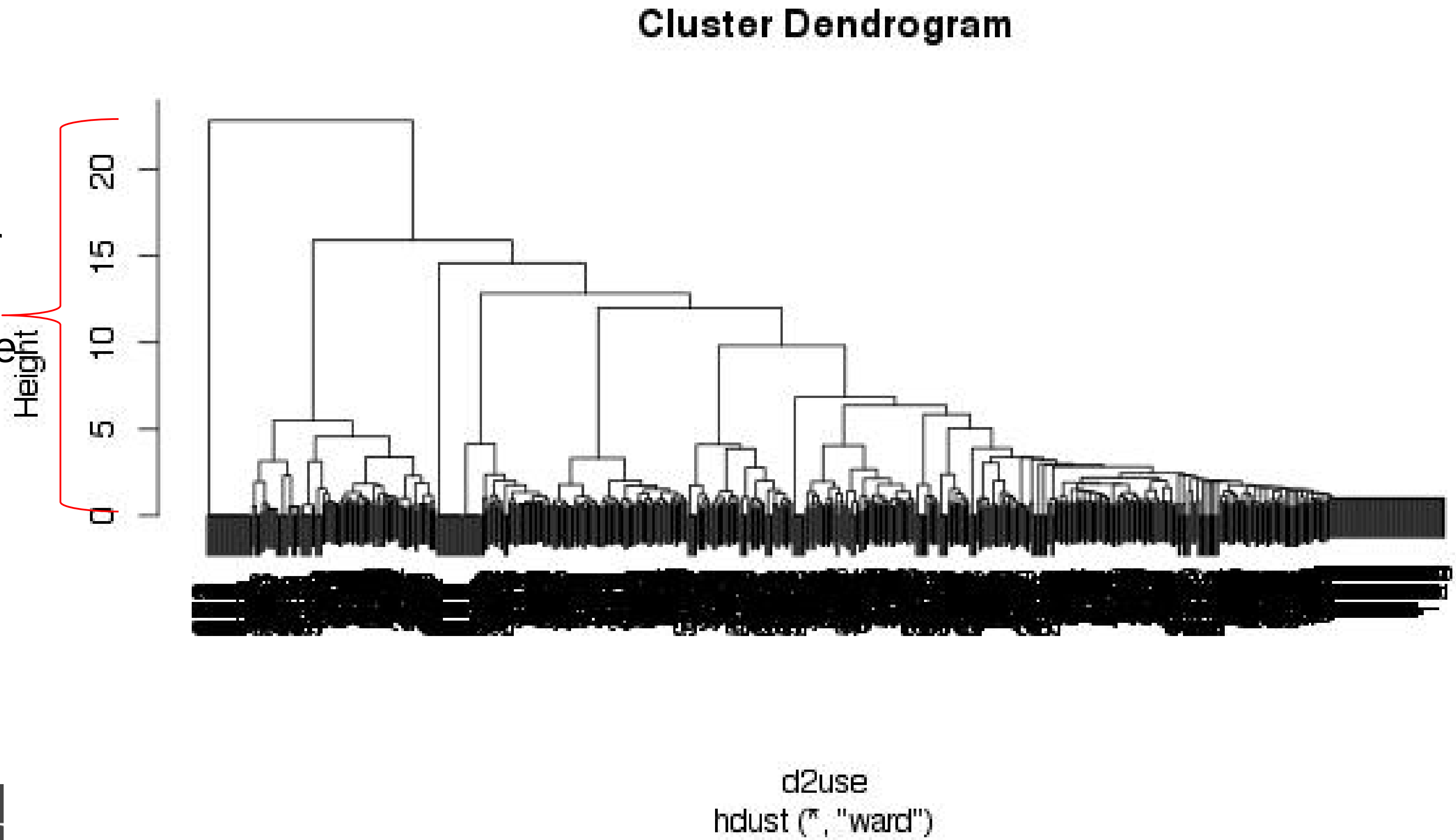


Other Clustering Methods

Hierarchical Clustering

- **hclust with "Ward" distance gives spherical clusters**

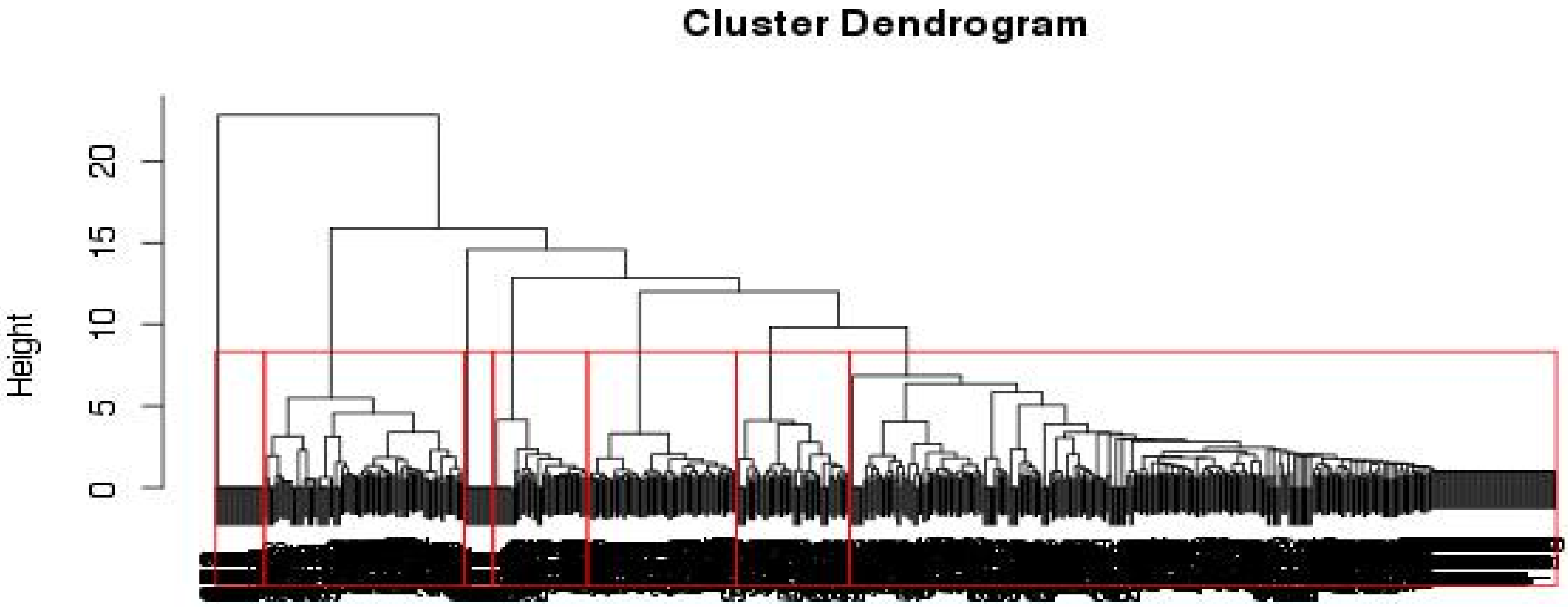
the cluster distance when 2 are combined



Hierarchical Clustering

- **Where height change looks big, cut off tree**

```
groups <- cutree(fit, k=7)
rect.hclust(fit, k=7, border="red")
```



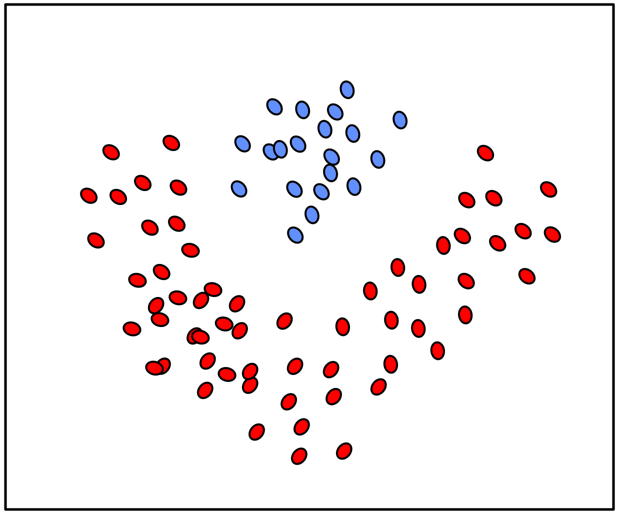
Other Clustering

- **Density based clustering**

  build neighborhoods around seed points

  link neighborhoods

  Results in arbitrary cluster shapes, good for image and spatial clustering



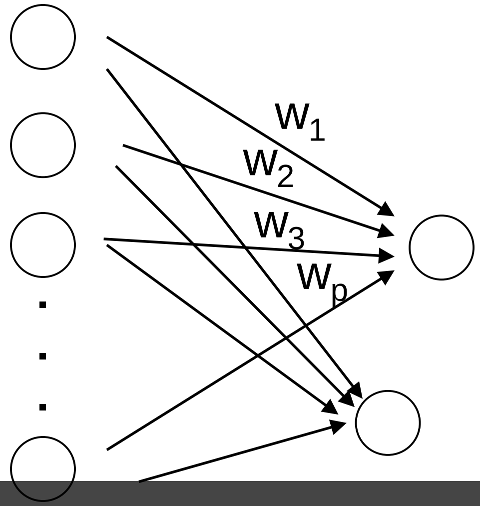
Other Clustering

- **Neural Network Based (e.g.)**

  initialize weights to coordinate values for a seed point

  set input nodes to data points

  get best match to seed for each data point and adjust weights toward the data point

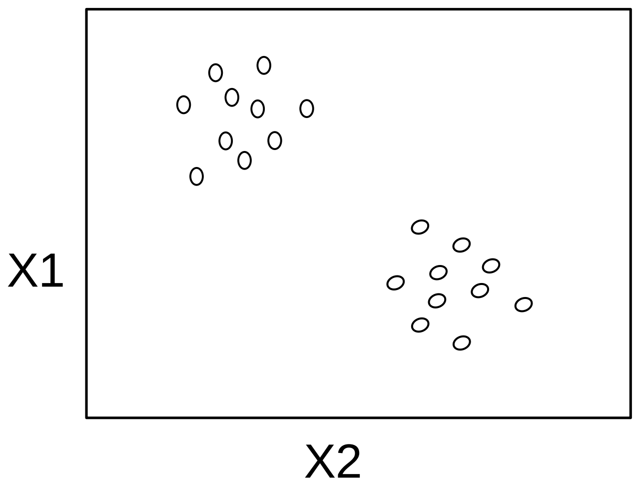


Target node(s), starts as a seed point and ends up as a cluster mean

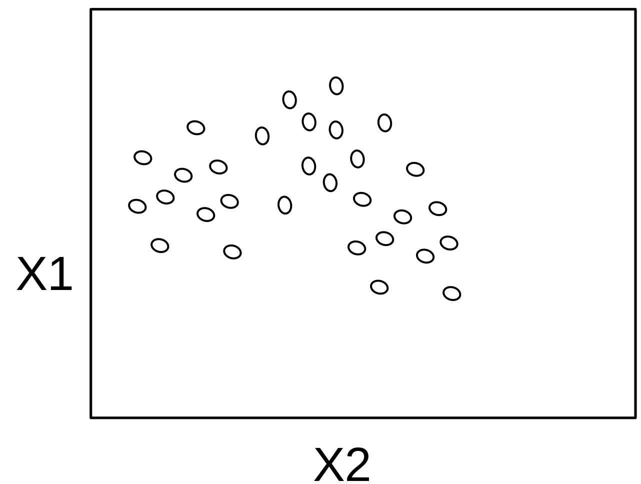
Pause

Imagine these 2 dimensional input spaces:
Which of these is easy or hard to cluster? (no class labels)

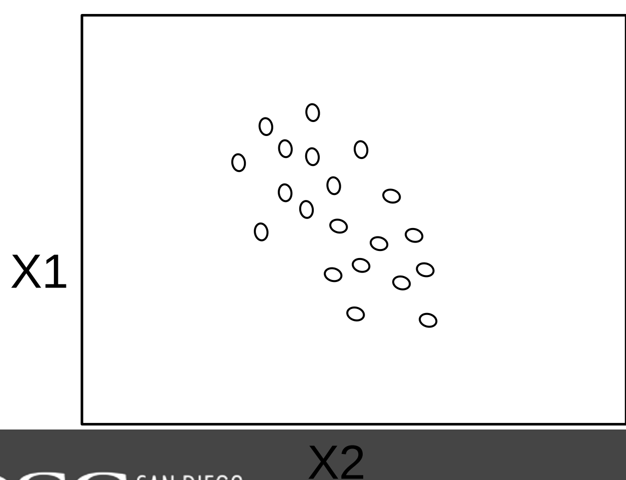
A



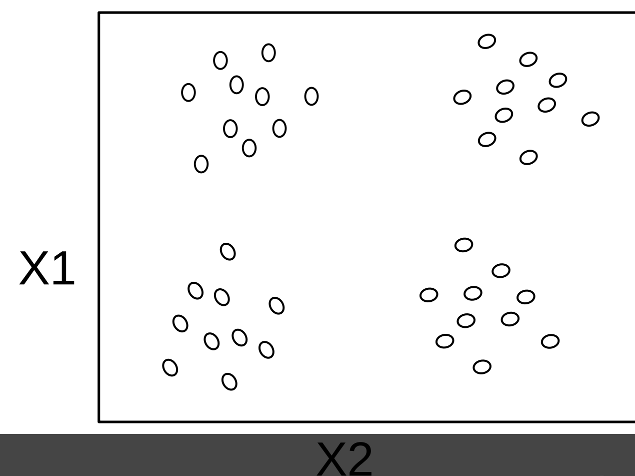
C



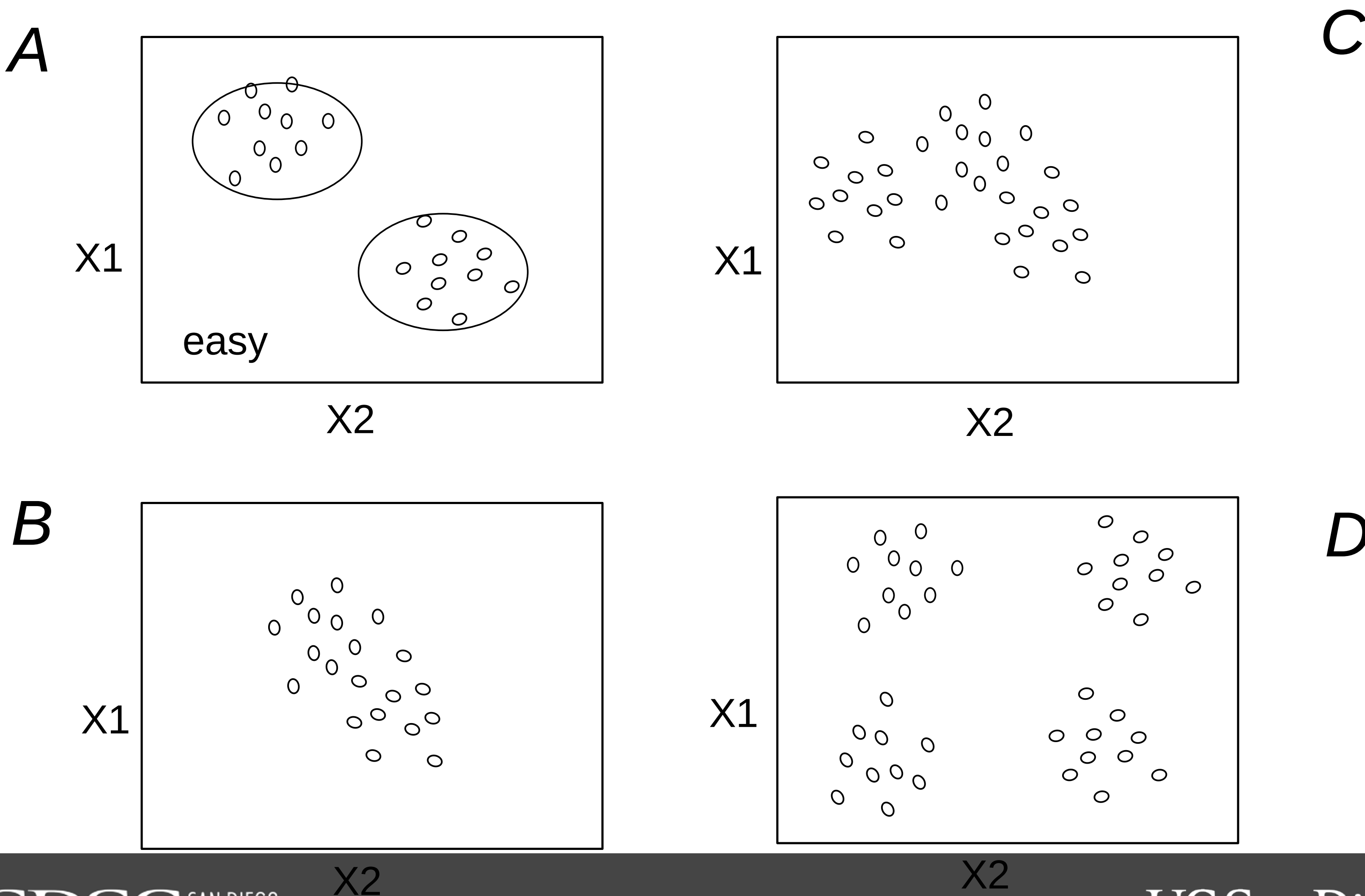
B



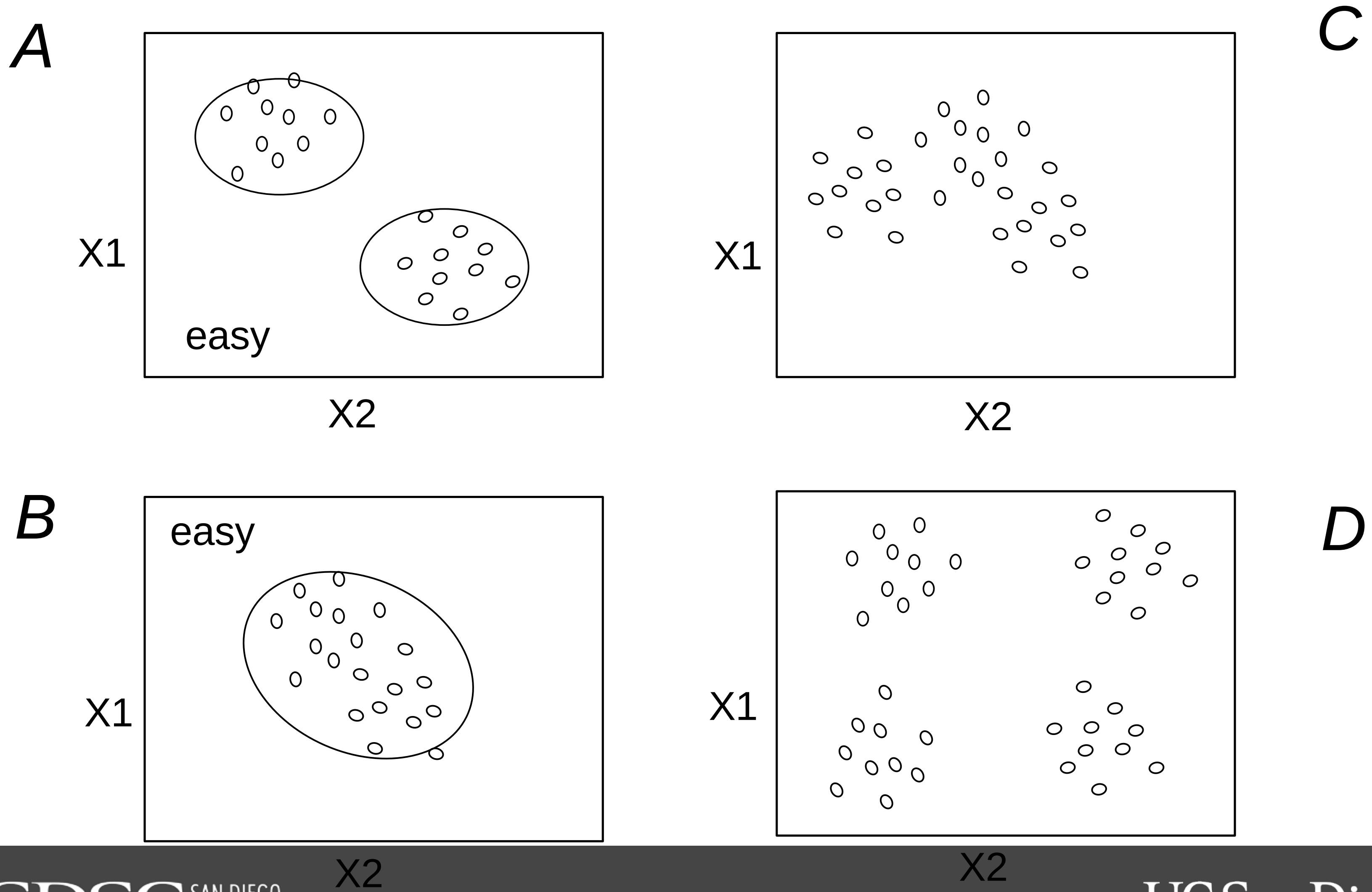
D



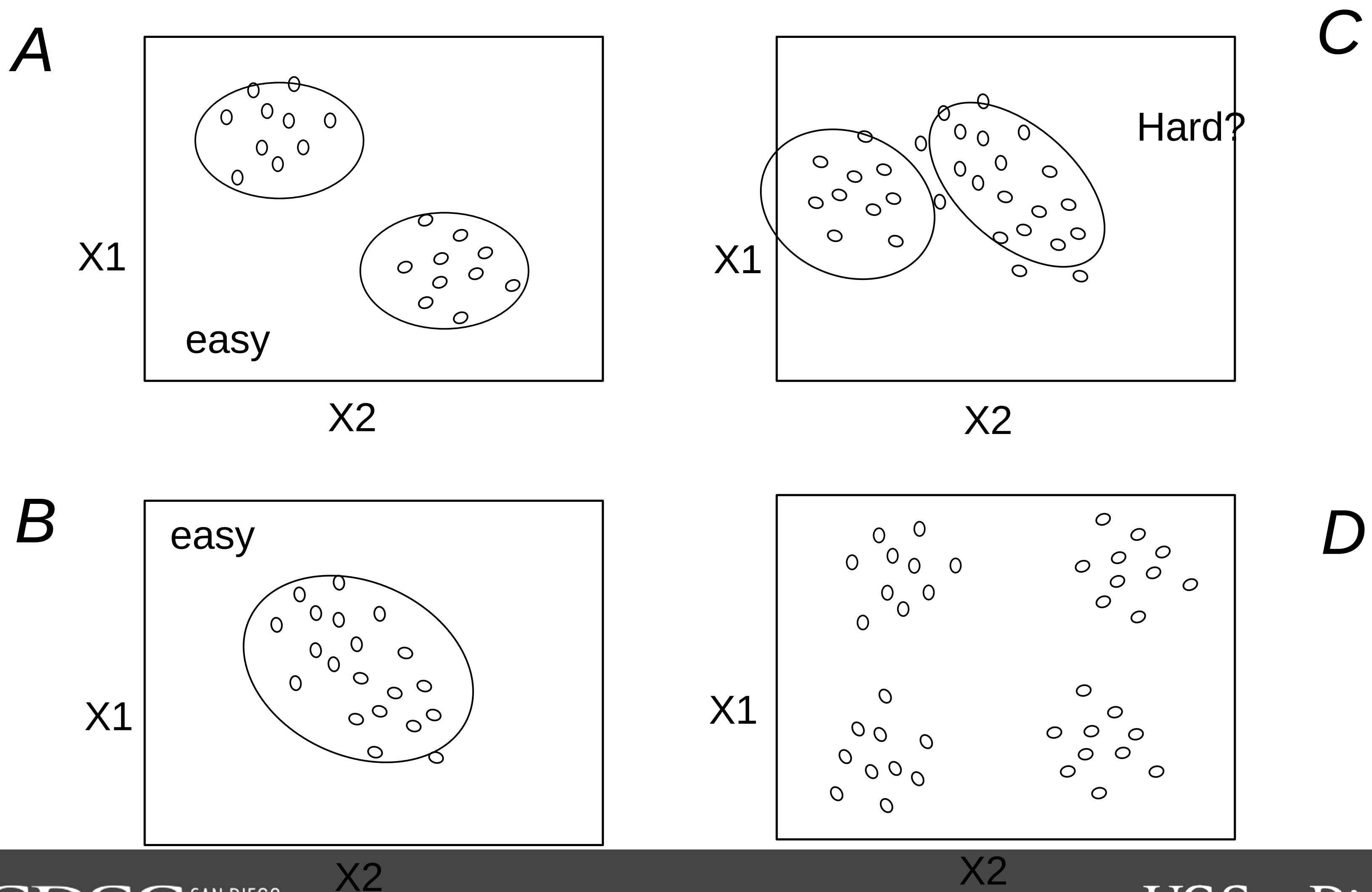
Potential clusters



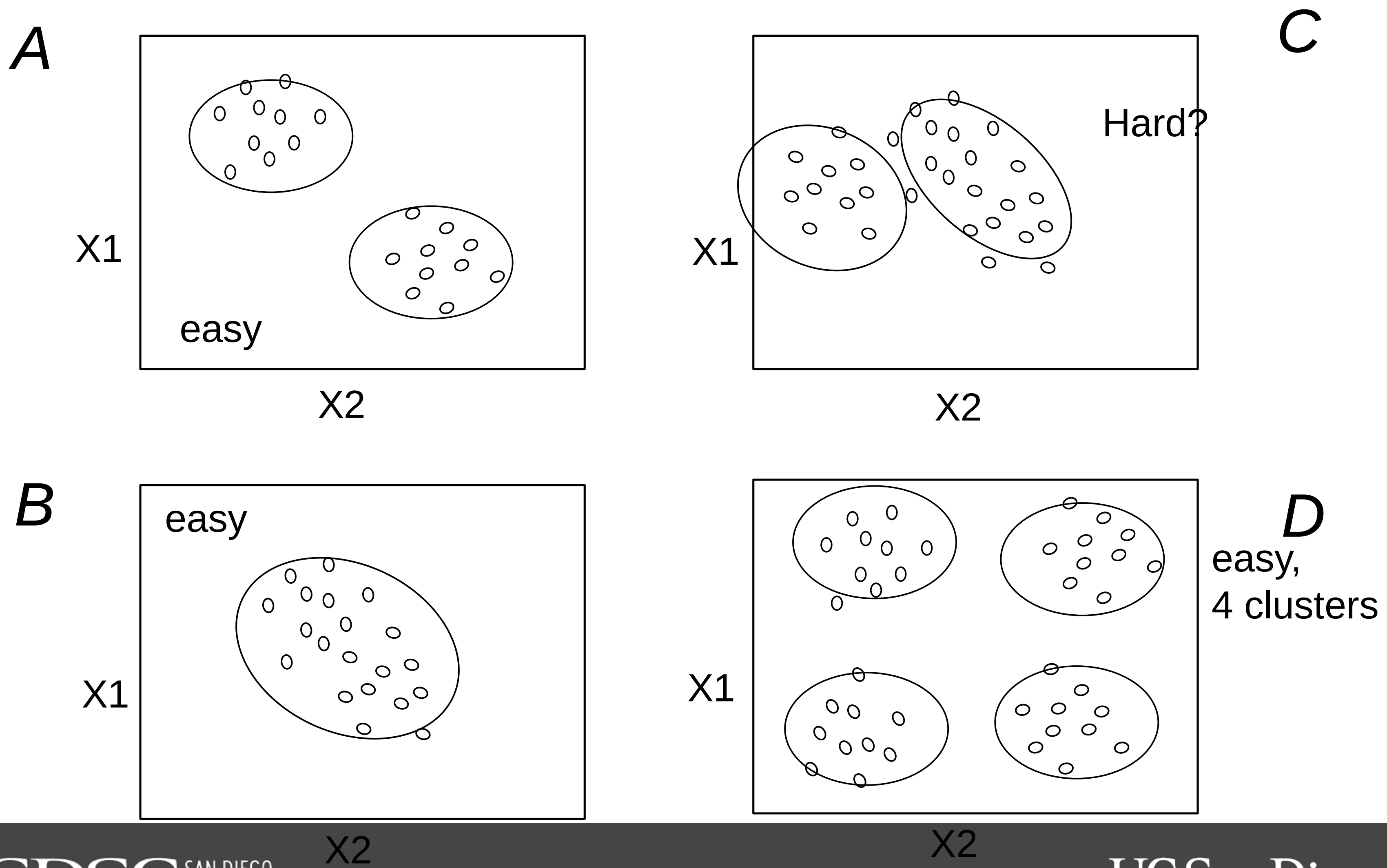
Potential clusters



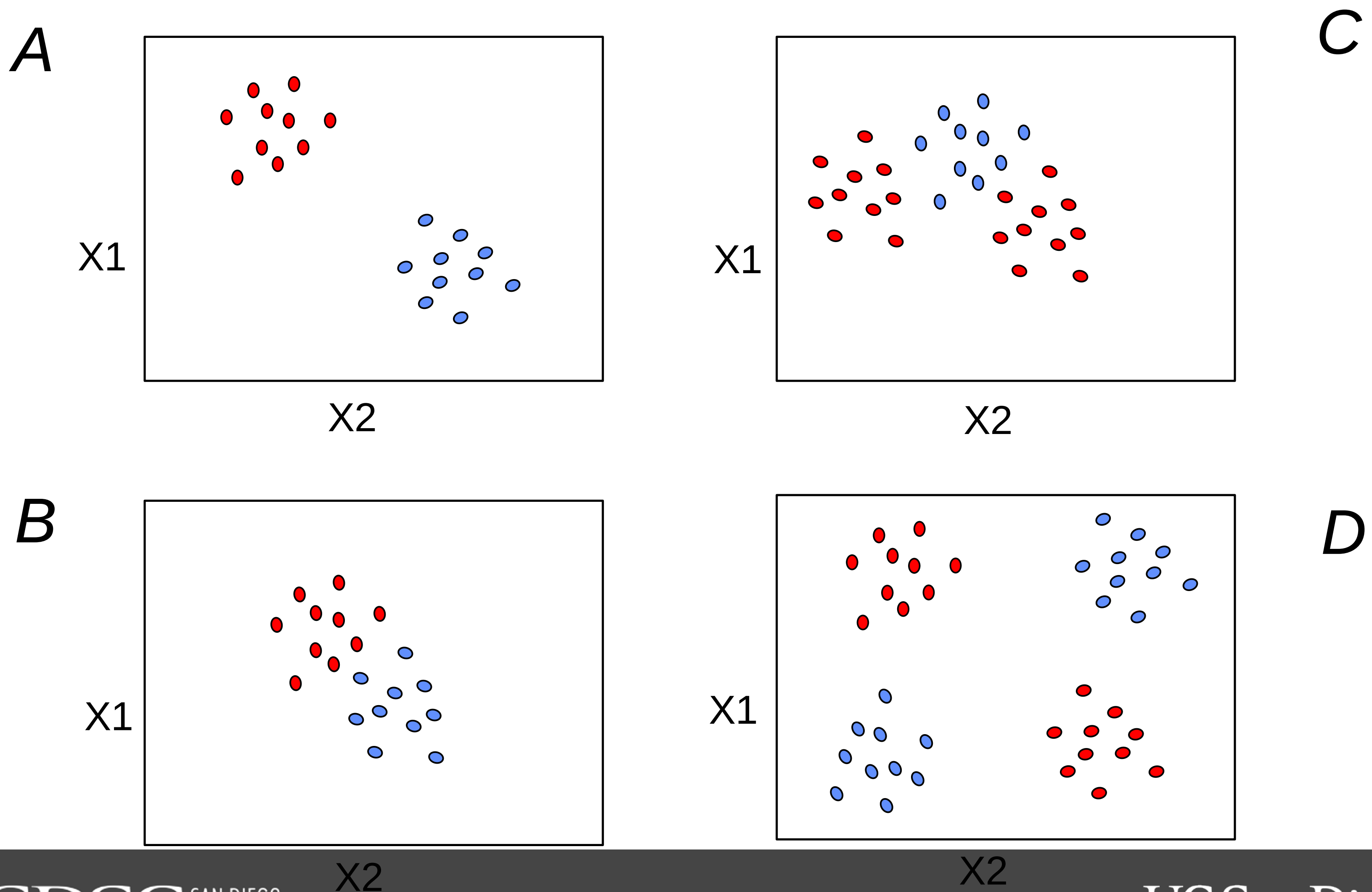
Potential clusters



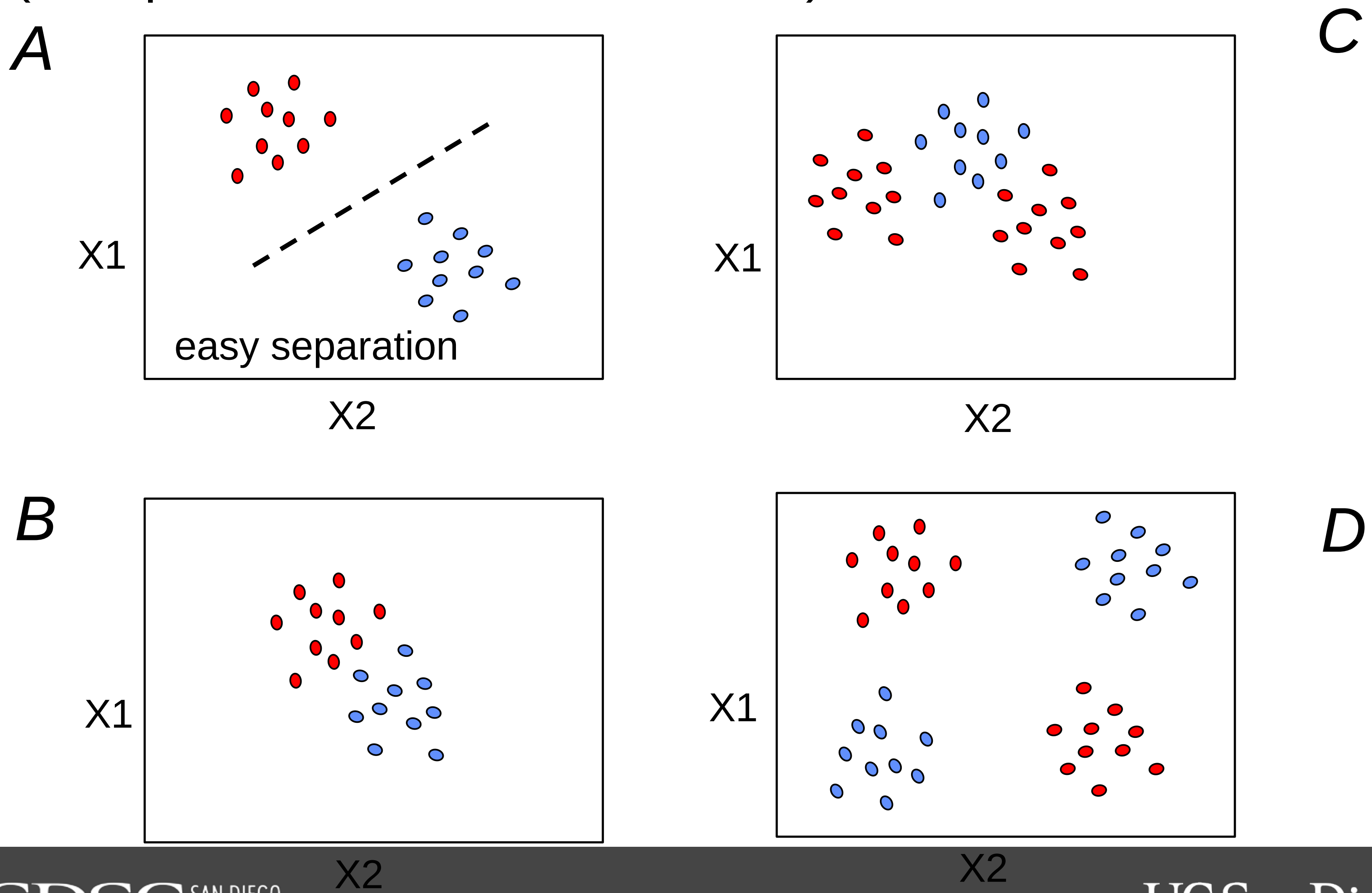
Potential clusters



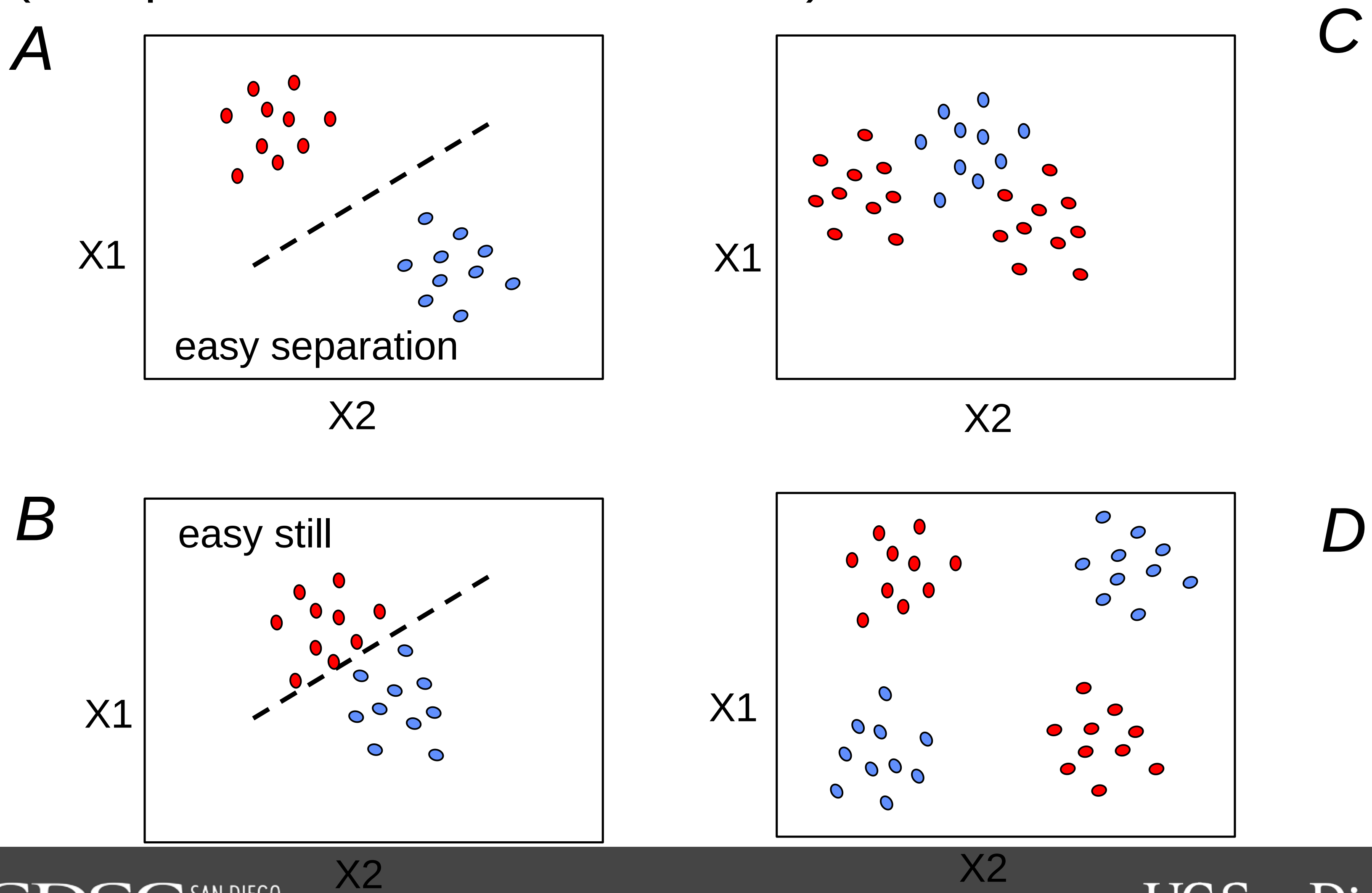
Now imaging there are two classes



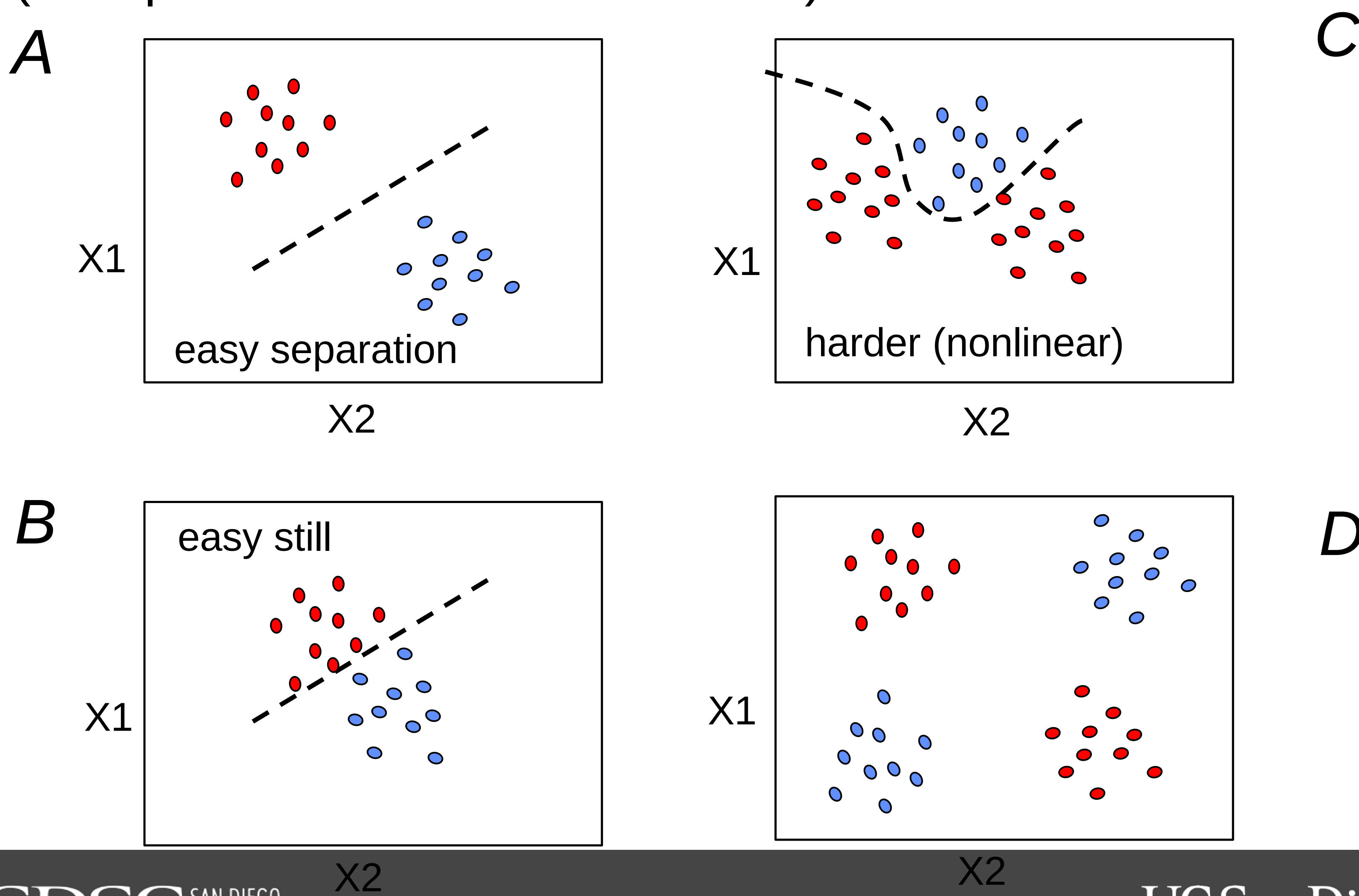
Which are easy or hard to classify?
(ie separate red or blue with lines)



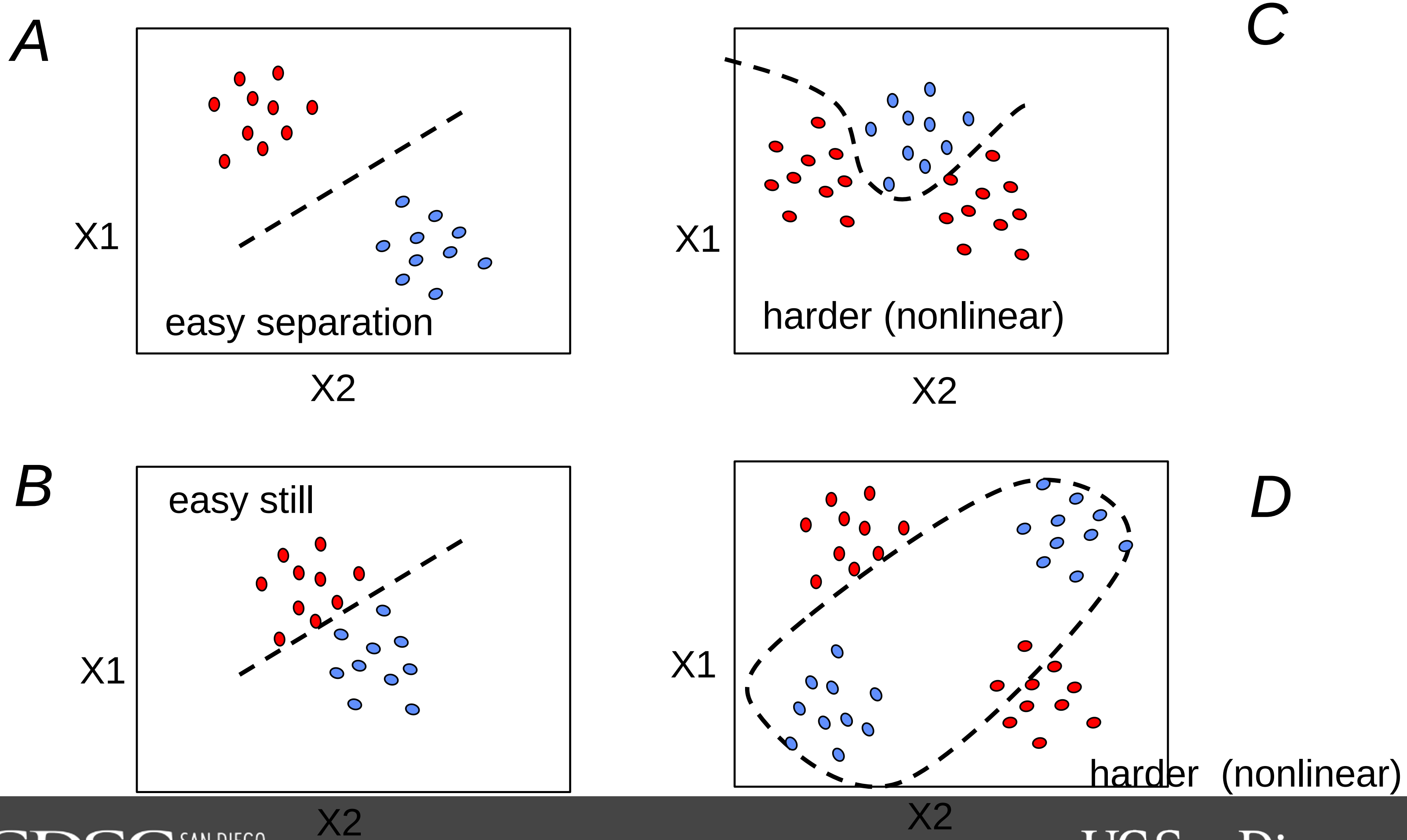
Which are easy or hard to classify?
(ie separate red or blue with lines)



Which are easy or hard to classify?
(ie separate red or blue with lines)

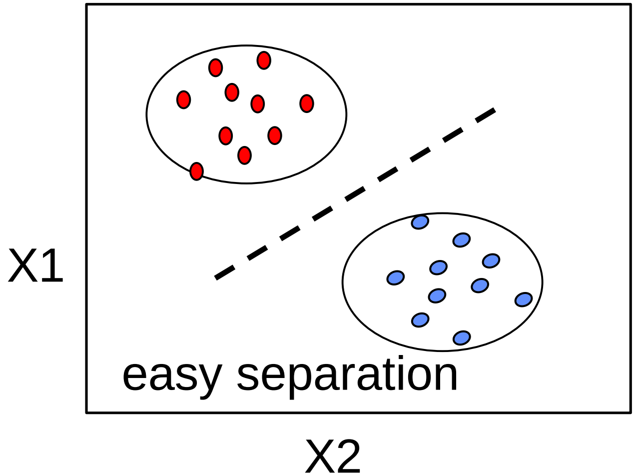


Which are easy or hard to classify?
(ie separate red or blue with lines)

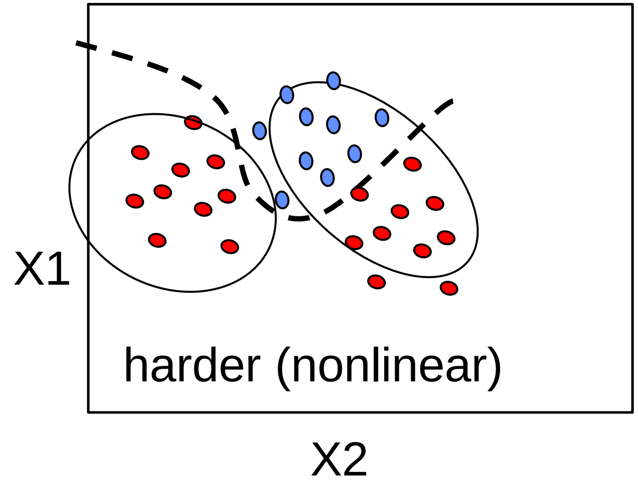


Which are easy or hard to classify?
(ie separate red or blue with lines)

A

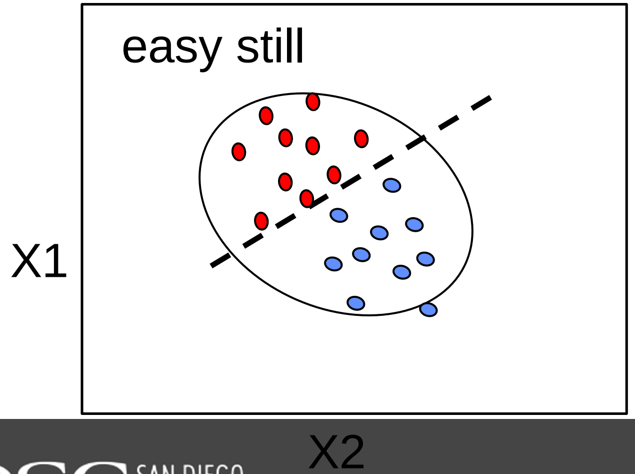


C

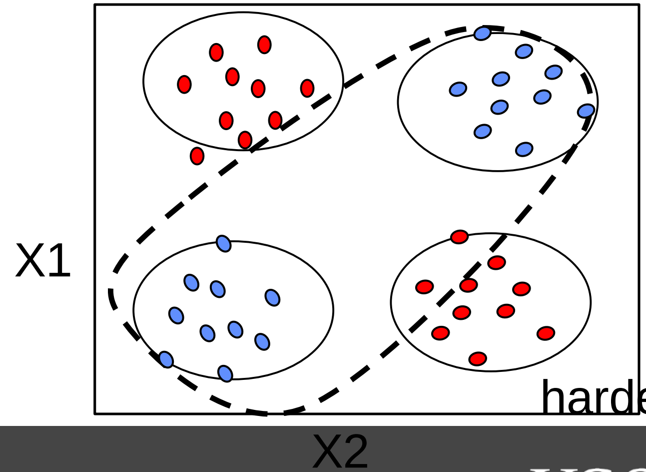


In summary:
No easy relationship between clusters and classification

B



D



harder (nonlinear)

Pause

Exercise, Kmeans and visualization in 2-D using SVD

- Using same data from SVD exercise, and SVD reduced matrix
- Run Kmeans
- Project data points onto first 2 SVD factors and plot them colored by cluster

(Which 2 factors?

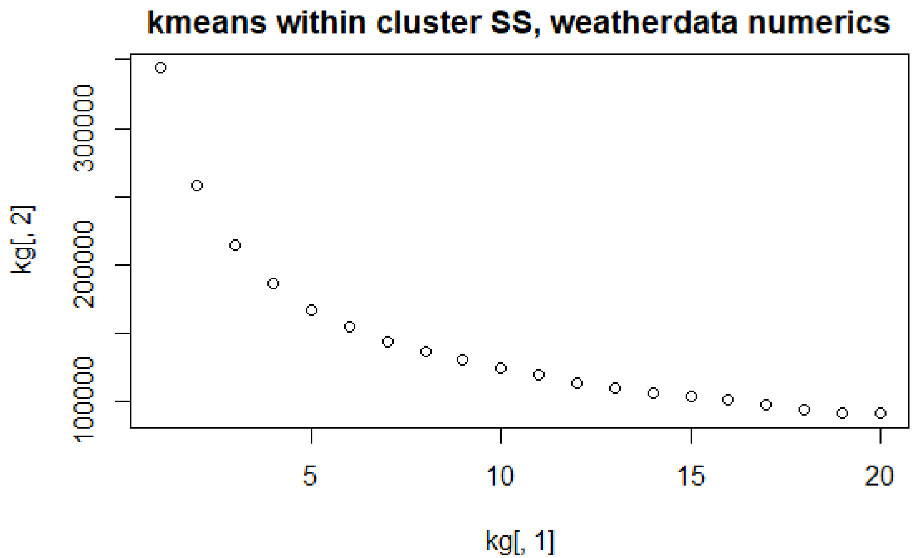
Keep in mind that:

Wsvd$U is N rows x P cols

Wsvd$V is PxP,

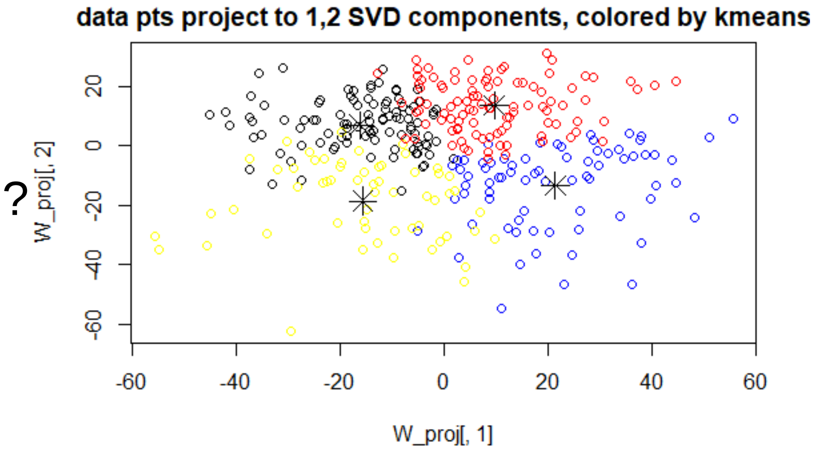
Data point is Px1 in original space and 2x1 in reduced space)

See clustering_exercise Rmd file
Rerun the SVD exercise if you need to, to get the W_mncntr and Wsvd matrices
Run Kmeans



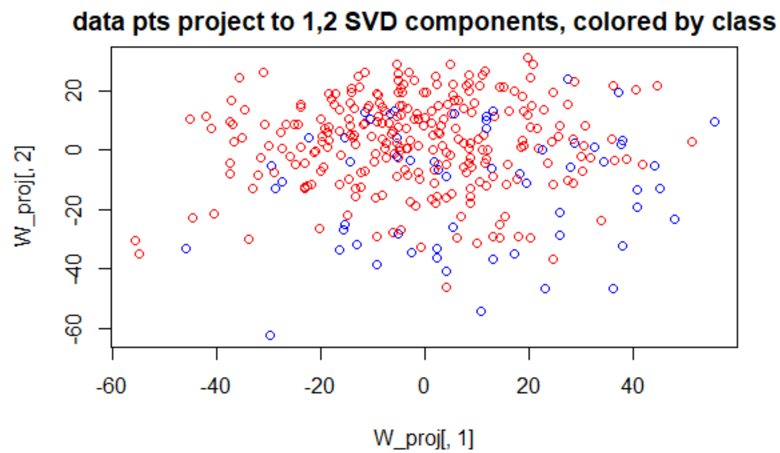
Run Kmeans with k=4
Plot data points onto 2-D space, use colors determined by Kmeans
Plot cluster centers

Are points well separated in 2-D projection?
Should they be?



Plot using colors determined by class (raintomorrow)

Are classes well separated?



Principle Components vs Clustering

- **PCA, SVD reduces dimensions, Clustering reduces to categorical groups**

- **In some cases, *k* PCs $\Leftrightarrow$ *k* clusters**

- **It is also useful to visualize clusters in PC space**

Summary

- Having no label doesn't stop you from finding structure in data

- Unsupervised methods are somewhat related