Supervised versus Unsupervised Learning

Supervised vs. Unsupervised Learning

- Machine learning problems can generally be categorized as supervised or unsupervised.
- The regression and classification problems that we have discussed so far are examples of supervised learning.
- What does it mean to be supervised? For each observation of the predictor measurement(s) x_i, $i = 1, 2, \cdots, N$, there is an *associated* response variable y_i.
- We wish to fit a model that relates the response to the predictors either for predicting the response based on future observations (prediction) or for understanding the relationship between the response and the predictors (inference).

Supervised vs. Unsupervised Learning (cont.)

- In contrast, unsupervised learning describes a more challenging scenario: We observe predictor variables x_i but there is no associated response variable y_i.

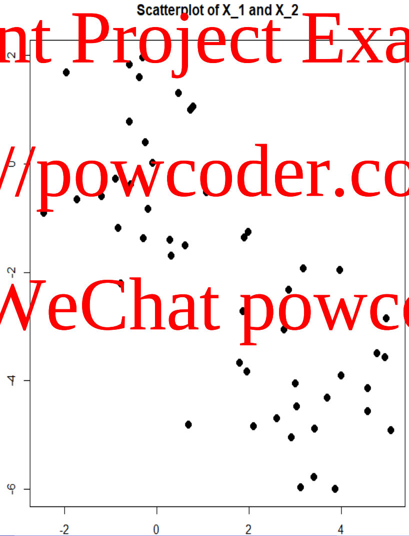
- It is not possible to fit a regression model or to do classification, since there is no response variable to predict.
- In this setting, we are working blind; the situation is referred to as unsupervised because we lack a response variable that can supervise our analysis.

Supervised vs. Unsupervised Learning (cont.)

- In statistics and machine learning, we week to find and understand relationships between variables, in this case of unsupervised learning, the relationship among the x_i's themselves (there are N many of them).
- One statistical learning tool that we can use here is cluster analysis, or clustering.
- The goal of cluster analysis is to ascertain, on the basis of $x_1, x_2, \cdots, x_N$, whether the observations fall into analysis relatively distinct groups.
- For example, in a market segmentation study, we might observe multiple characteristics (variables) for potential customers, such as zip code, family income, and shopping habits. We might believe that the customers fall into different groups, such as big spenders versus low spenders.

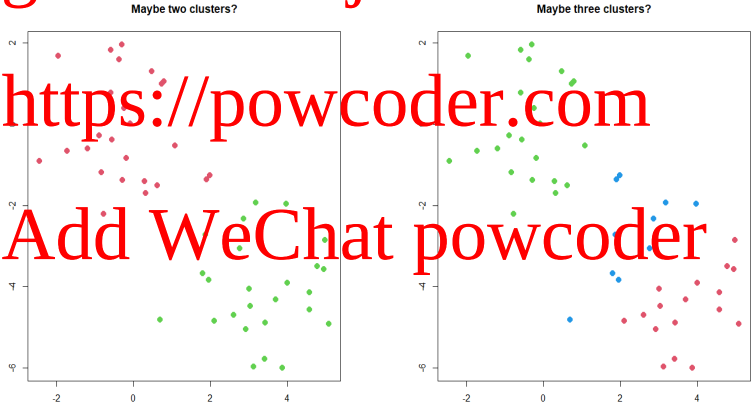
Supervised vs. Unsupervised Learning (cont.)

- The following figure provides a simple illustration of the clustering problem. It is a plot of 50 measurements on two variables (X_1, X_2).



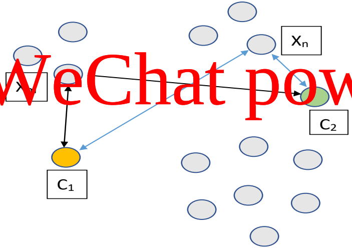
Supervised vs. Unsupervised Learning (cont.)

- We wish to see how well the observations are clustered into groups and how well the groups are separated between each other.
- But the problem is do we cluster into 2 or 3 or more groups? How do we determine this?



Idea and Algorithm of K-means Clustering

- Suppose we have N d-dimensional observations $x_i \equiv (x_{i1}, x_{i2}, \cdots, x_{id}) \in R^d$, $i = 1, 2, \cdots, N$.
- The k-means clustering algorithm requires a predetermined choice for the number of centroids of the clusters, say K.
- Given K, randomly initialize the K centroids, say $C_1, C_2, \cdots, C_K$, where $C_k \equiv (C_{k1}, C_{k2}, \cdots, C_{kd}) \in R^d$, for $k = 1, 2, \cdots, K$.
- Given $C_1, C_2, \cdots, C_K$, compute the distance of x_i from each centroid C_k.



K-means (cont.)

- Mathematically, this means we compute the distance Δ_i corresponding to observation x_i defined by

$$\Delta_i = \min_{1 \le k \le K} \sum_{j=1}^{d} (x_{ij} - C_{kj})^2$$

and assign x_i to cluster k^* if Δ_i achieves its minimum for $k = k^*$. This is the STEP I.

- STEP I: Cluster assignment: The cluster label assignment for x_i is taken as $L(x_i) = k^*$ where

$$k^* = \arg\min_{1 \le k \le K} \sum_{j=1}^{d} (x_{ij} - C_{kj})^2$$

K-means (cont.)

- STEP II: Given the cluster assignment for each x_i, recalculate the new centroids as

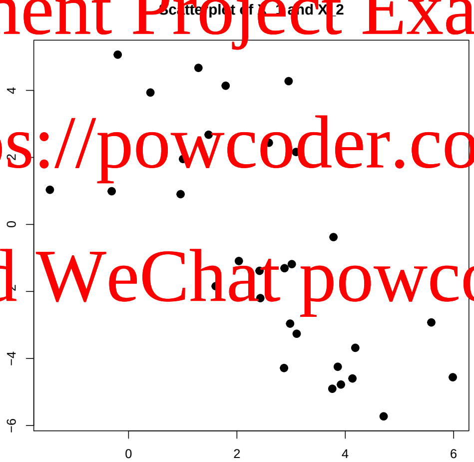
$$C_k = \frac{1}{n_k} \sum_{i: L(x_i)=k} x_i$$

where n_k is the number of observations x_i with $L(x_i) = k$.

- Cycle between STEPS I and II, that is, the cluster assignment and centroid computation, until convergence.

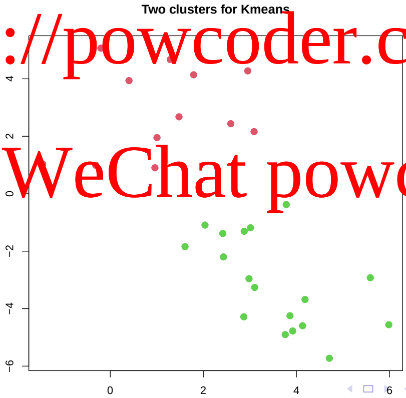
R codes

```
plot(x, col="black", main="Scatterplot of X_1 and X_2",
 xlab="", ylab="", pch=20, cex=2)
```

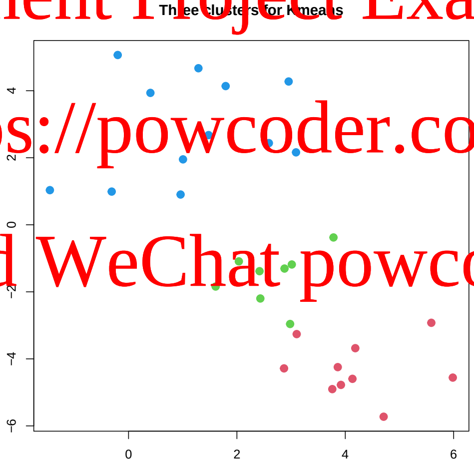
R codes (cont.)

```r
#Start with two clusters
km.out=kmeans(x,2,nstart=20)
plot(x, col=(km.out$cluster+1),
 main="Two clusters for Kmeans",
 xlab="", ylab="", pch=20, cex=2)
```

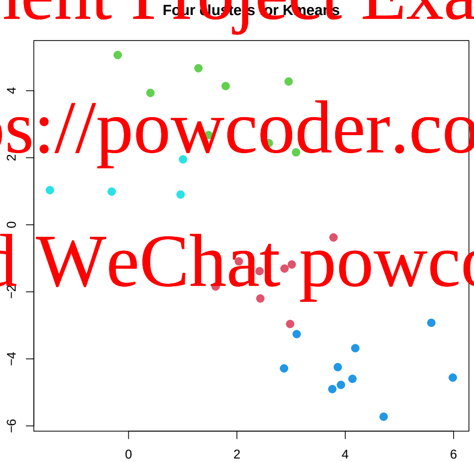
R codes (cont.)

```
#Three clusters
km.out=kmeans(x,3,nstart=20)
```

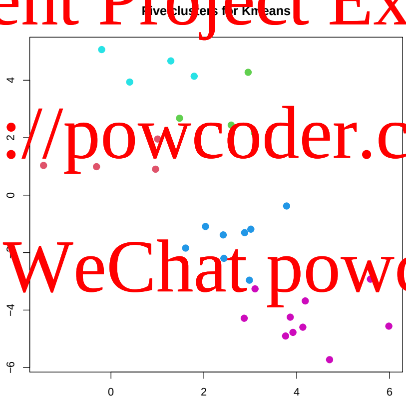
R codes (cont.)

```
#Four clusters
km.out=kmeans(x,4,nstart=20)
```

R codes (cont.)

```
#Five clusters
km.out=kmeans(x,5,nstart=20)
```

Where to stop? Which is the optimal cluster number K?

Choosing an optimal K

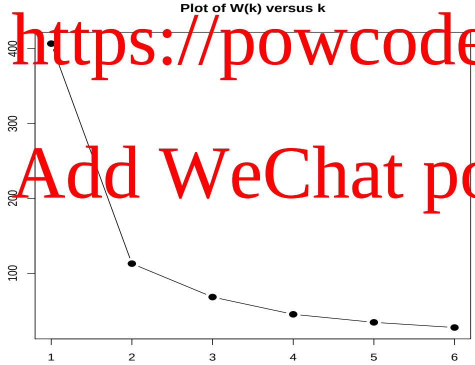
- K is like the flexibility parameter. The larger the value of K, the kmeans clustering algorithm "fits" the data better and better.
- The maximum value of $K = N$ results in each observation x_i being in its own cluster and centroid.
- To avoid overfitting, consider the within-cluster-variance

$$W = \sum_{k=1}^{K} \sum_{i:L(x_i)=k} \sum_{j=1}^{d} (x_{ij} - c_{kj})^2$$

  Note that as $K \uparrow N$, $W \downarrow 0$.
- As K increases, the drop in W will be large initially until a point where subsequent drops in W will no longer be that significant.
- We choose the optimal K, K^*, as the point where the decrease in W starts to become insignificant. This criteria of choosing K^* is called the elbow criteria.

R codes for optimal K^*

```
#Choosing optimal K
Kmax = 6;W <- vector("numeric", Kmax)
for (k in 1:Kmax){W[k] = kmeans(x,k,nstart=20)$tot.withinss}
plot(c(1:Kmax), W, main="Plot of W(k) versus k",
 type="b", xlab="", ylab="", pch=20, cex=2)
```

Seems like $K^* = 3$.

Hard versus Soft Thresholding for Clustering

- Kmeans is a hard thresholding clustering procedure. Each observation is clustered into one and only one cluster according to

$$k^* = \arg \min_{1 \leq k \leq K} \sum_{j=1}^{d} (x_{ij} - C_{kj})^2$$

- However, observations located at an equidistant point between two centroids should not be clustered into one or the other.
- Soft thresholding means that there is a probability associated with each observation of being included into each cluster.

- For observations lying at an equidistant point between two centroids should have probabilities 0.5 and 0.5 of belonging to each cluster.
- How to achieve this?

Gaussian Mixture Models

- The Gaussian mixture model (GMM) is a model based clustering teachnique that uses a likelihood based approach to estimate the probability of each observation belonging to a particular cluster.
- The GMM assumes that the observations belong to each class is normally distributed density function with unknown mean and variance.
- The GMM likelihood with K components looks like this:

$$\pi(x) = \sum_{k=1}^{K} p_k \, \phi_d(x\,;\, \mu_k, \Sigma_k)$$

  where $p_k \geq 0$ are mixture probabilities summing to 1, $\phi_d(x\,;\,\mu, \Sigma)$ is the d-variate normal pdf with mean μ and covariance matrix Σ.
- The mixture probabilities p_k, $k = 1, 2, \cdots, K$ and the means and covariance, μ_k and Σ_k, of each class k, $k = 1, 2, \cdots, K$ are unknown and have to be estimated based on a training dataset.

Selecting the optimal K^*

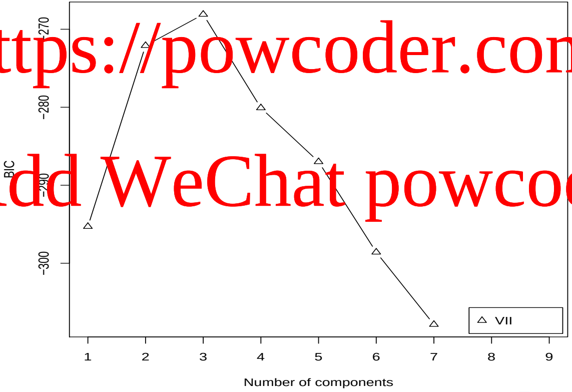
- The optimal K^* is determined using the Bayes Information Criteria (BIC). This is a model selection tool that penalizes overfitting.
- The formula for the BIC is

$$BIC(K) = -2 \log \hat{\ell}_K + m_K \log(N)$$

  where m_K is the total number of unknown parameters for a GMM with K components, $\hat{\ell}_K$ is the estimate of the GMM likelihood function based on parameter values learned from a training dataset of size N.
- This should be a positive number and the optimal K^* will minimize the BIC. However the R package calls $-BIC$ as the BIC, so it will find the optimal K^* by maximizing its version of BIC.

R codes for GMM fitting

```
#GMM fitting
library(mclust)
BIC <- mclustBIC(x, modelNames = "VII")
#Determine K using the BIC criteria
plot(BIC)
```

```r
summary(BIC)

## Best BIC values:
##                VII,3       VII,2       VII,4
## BIC      -268.0977 -272.09910 -280.08661
## BIC diff    0.0000   -4.00136  -11.98887

mod1 <- Mclust(x, x = BIC)
summary(mod1, parameters = TRUE)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 3 compone
##
##  log-likelihood  n df       BIC       ICL
##       -115.3423 30 11 -268.0977 -269.0544
```

```
## Clustering table:
##  1  2  3
##  7 11 12
##
## Mixing probabilities:
##         1         2         3
## 0.2375402 0.3583167 0.4041431
##
## Means:
##           [,1]      [,2]     [,3]
## [1,]  2.596463  4.123592 1.157286
## [2,] -1.416490 -4.198373 2.818566
##
```

```
## Variances:
## [,,1]
##          [,1]     [,2]
## [1,] 0.388412 0.000000
## [2,] 0.000000 0.388412
## [,,2]
##           [,1]      [,2]
## [1,] 0.8079786 0.0000000
## [2,] 0.0000000 0.8079786
## [,,3]
##          [,1]     [,2]
## [1,] 1.991936 0.000000
## [2,] 0.000000 1.991936
```

R codes for classification labels

```
plot(mod1, what = "classification")
```