# Data Mining (W4240 Section 001)
# Clustering (part 1)

Giovanni Motta

Columbia University, Department of Statistics

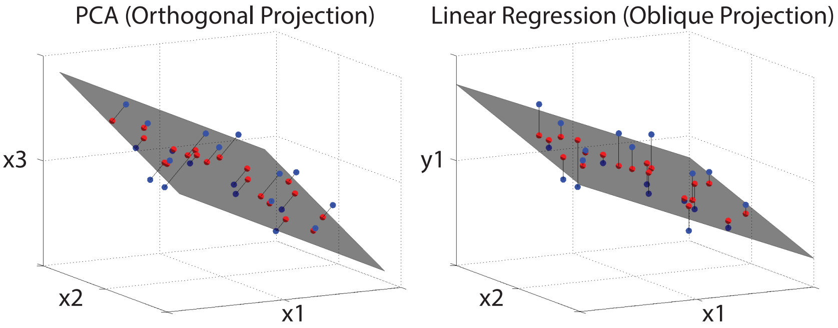
November 30, 2015

# Outline

# Outline

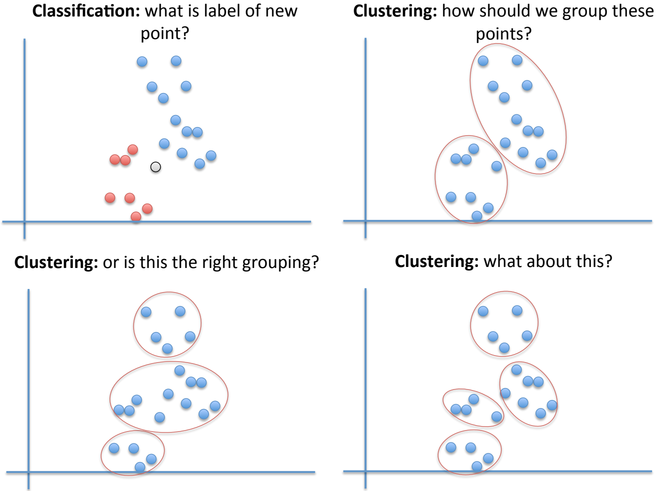
# PCA vs Linear Regression

- Unsupervised learning seeks explanatory factors
- Supervised learning asserts explanatory factors



# Today: Clustering

The essential difference here is again unsupervised/supervised:



# Outline

# K-Means

$$\text{Sample space } = \overset{\text{质心}}{C_1 \cup C_2} \cup \cdots \cup C_K, \text{ with } C_k \cap C_j = \emptyset$$

K-means is the simplest clustering method available.

Start with a notion of distance between each pair of points (Euclidean, 0-1 loss, some combination thereof)

Each cluster $k$ has a *centroid*, or average value $\mu_k$

Example: data for cluster $k$ is $(0, 1)$, $(0.5, 0.5)$, $(1, 1)$

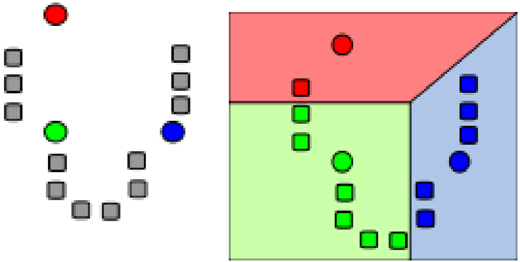
$$\mu_{k1} = \frac{1}{3}(0 + 0.5 + 1) = \frac{1}{2}$$

$$\mu_{k2} = \frac{1}{3}(1 + 0.5 + 1) = \frac{5}{6}$$

# K-Means

质心

Once we have a set of centroids, $\mu_1, \ldots, \mu_K$, we can ask which data are closest to the centroids[1]



Note: the regions have linear boundaries

[1] Photo credit: Wikipedia

# K-Means

K-Means迭代算法 **`First of all, SCALE THE DATA!!!`**

Conceptually, we fit $K$ clusters with the following steps:

1. pick $K$ initial cluster means
2. associate all points closest to mean $k$ with cluster $k$
3. use points in cluster $k$ to update mean for that cluster
4. re-associate points closest to new mean for $k$ with cluster $k$
5. use new points in cluster $k$ to update mean for that cluster
6. ...
7. stop when no change between updates

# K-Means

Within-Clustering Variation (WCV):

$$WCV = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,\ell \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{\ell j})^2$$

Residual Sum of Squares (RSS):

$$\begin{aligned} RSS &= \sum_{k} \sum_{i:C_i=k} \underline{d(\mathbf{x}_i, \boldsymbol{\mu}_k)} \\ &= \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \underline{\mu_{kj}})^2 \end{aligned}$$

The K-means algorithm is guaranteed to decrease the WCV since

$$\frac{1}{|C_k|} \sum_{i,\ell \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{\ell j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \underline{\overline{x}_{kj}})^2,$$

where $\overline{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$.

K-means:

1. Minimize RSS over cluster assignments $C_i$:

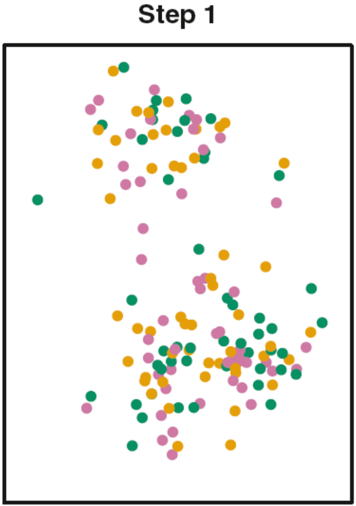
$$\arg\min_{C_i} \sum_{k=1}^{K} \sum_{i:C_i=k} \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2$$

2. Minimize RSS over cluster centroids $\mu_k$:

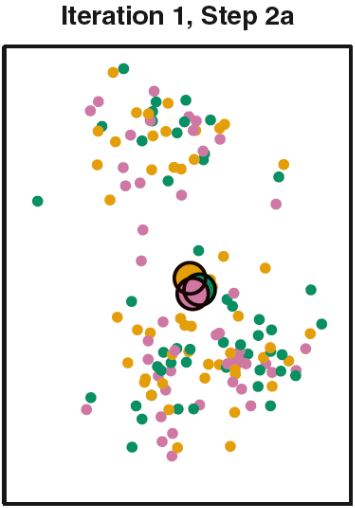
$$\arg\min_{\mu_k} \sum_{k=1}^{K} \sum_{i:C_i=k} \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2$$

# K-Means: Observations

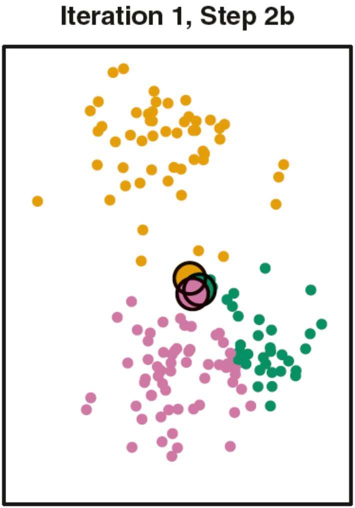
# K-Means: observation randomly assigned to a cluster



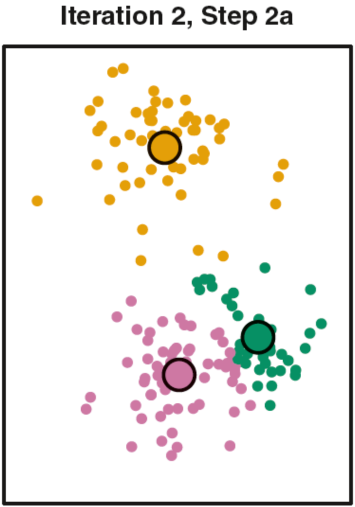
# K-Means: cluster centroids



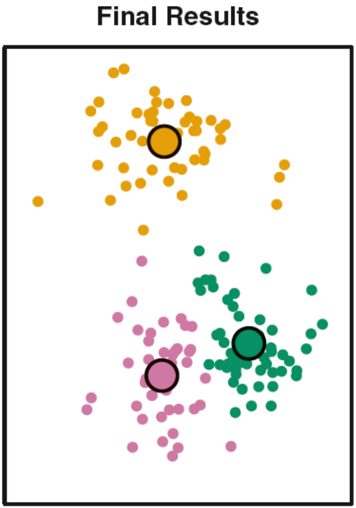
# K-Means: each obs is assigned to the nearest centroid



# K-Means: new cluster centroids



# K-Means: results after 10 iterations



# K-Means

Will K-means converge? Can't it just limit cycle?

- reassignment: each observation moves to closest centroid
- update: new centroid minimizes RSS for this assignment

$$RSS_k(\mu) = \sum_{i\,:\,C_i=k} \sum_{j=1}^{d} (x_{ij} - \mu_{kj})^2$$

$$\frac{\partial}{\partial \mu_j} RSS_k(\mu) = \sum_{i\,:\,C_i=k} 2(x_{ij} - \mu_{kj})$$

$$\mu_{kj} = \frac{1}{n_k} \sum_{i\,:\,C_i=k} x_{ij}$$

- Will the final labels and means always be the same?

# K-Means

Will K-means converge? Can't it just limit cycle?

- reassignment: each observation moves to closest centroid
- update: new centroid minimizes RSS for this assignment

$$RSS_k(\mu) = \sum_{i\,:\,C_i=k} \sum_{j=1}^{d} (x_{ij} - \mu_{kj})^2$$

$$\frac{\partial}{\partial \mu_j} RSS_k(\mu) = \sum_{i\,:\,C_i=k} 2(x_{ij} - \mu_{kj})$$

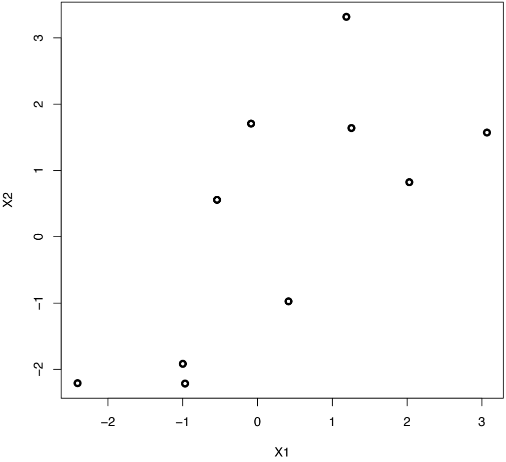
$$\mu_{kj} = \frac{1}{n_k} \sum_{i\,:\,C_i=k} x_{ij}$$

- Will the final labels and means always be the same?
- No. K-means has (many) local optima.

# K-Means: Example

Data:

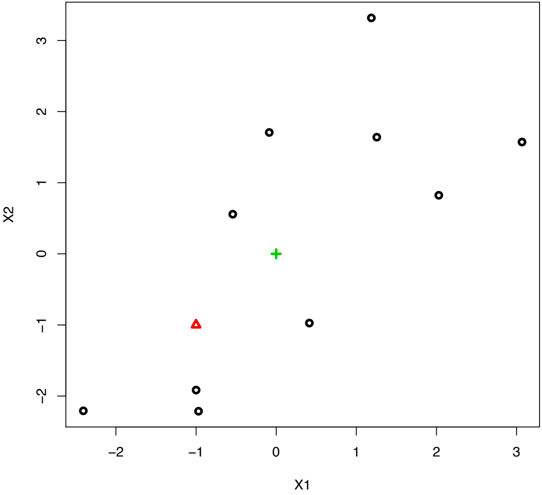
| $x_1$ | $x_2$ |
|---|---|
| 0.4 | -1.0 |
| -1.0 | -2.2 |
| -2.4 | -2.2 |
| -1.0 | -1.9 |
| -0.5 | 0.6 |
| -0.1 | 1.7 |
| 1.2 | 3.3 |
| 3.1 | 1.6 |
| 1.3 | 1.6 |
| 2.0 | 0.8 |



# K-Means: Example

Pick $K$ centers (randomly):

$$(-1, -1) \text{ and } (0, 0)$$



# K-Means: Example

Calculate distance between points and those centers:

| $x_1$ | $x_2$ | $(-1,-1)$ | $(0,0)$ |
|---|---|---|---|
| 0.4 | -1.0 | 1.4 | 1.1 |
| -1.0 | -2.2 | 1.2 | 2.4 |
| -2.4 | -2.2 | 1.9 | 3.3 |
| -1.0 | -1.9 | 0.9 | 2.2 |
| -0.5 | 0.6 | 1.6 | 0.8 |
| -0.1 | 1.7 | 2.9 | 1.7 |
| 1.2 | 3.3 | 4.8 | 3.5 |
| 3.1 | 1.6 | 4.8 | 3.4 |
| 1.3 | 1.6 | 3.5 | 2.1 |
| 2.0 | 0.8 | 3.5 | 2.2 |

```
> centers <- rbind(c(-1,-1),c(0,0))
> dist1 <- apply(x,1,function(x) sqrt(sum((x-centers[1,])^2)))
> dist2 <- apply(x,1,function(x) sqrt(sum((x-centers[2,])^2)))
```

# K-Means: Example

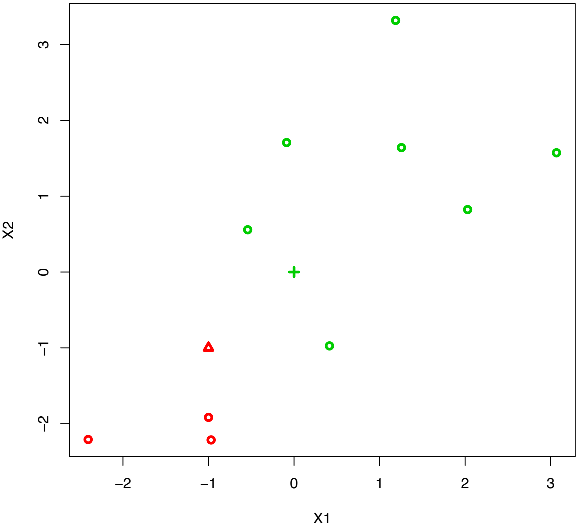
Choose mean with smaller distance:

| $x_1$ | $x_2$ | $(-1,-1)$ | $(0,0)$ |
|---|---|---|---|
| 0.4 | -1.0 | 1.4 | **1.1** |
| -1.0 | -2.2 | **1.2** | 2.4 |
| -2.4 | -2.2 | **1.9** | 3.3 |
| -1.0 | -1.9 | **0.9** | 2.2 |
| -0.5 | 0.6 | 1.6 | **0.8** |
| -0.1 | 1.7 | 2.9 | **1.7** |
| 1.2 | 3.3 | 4.8 | **3.5** |
| 3.1 | 1.6 | 4.8 | **3.4** |
| 1.3 | 1.6 | 3.5 | **2.1** |
| 2.0 | 0.8 | 3.5 | **2.2** |

```
> dists <- cbind(dist1,dist2)
> cluster.ind <- apply(dists,1,which.min)
```

# K-Means: Example

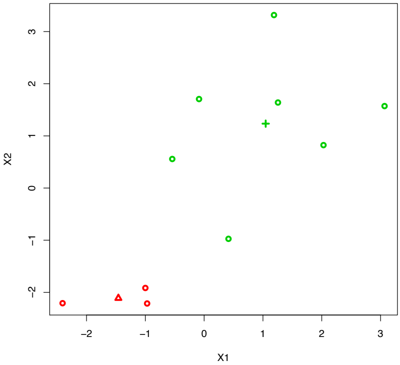
New clusters:



# K-Means: Example

Refit means for each cluster:

- cluster 1: $(-1.0, -2.2)$, $(-2.4, -2.2)$, $(-1.0, -1.9)$
- new mean: $(-1.5, -2.1)$
- cluster 2: $(0.4, -1.0)$, $(-0.5, 0.6)$, $(-0.1, 1.7)$, $(1.2, 3.3)$, $(3.1, 1.6)$, $(1.3, 1.6)$, $(2.0, 0.8)$
- new mean: $(1.0, 1.2)$



# K-Means: Example

Recalculate distances for each cluster:

| $x_1$ | $x_2$ | $(-1.5, -2.1)$ | $(1.0, 1.2)$ |
|---|---|---|---|
| 0.4 | -1.0 | 2.2 | 2.3 |
| -1.0 | -2.2 | 0.5 | 4.0 |
| -2.4 | -2.2 | 1.0 | 4.9 |
| -1.0 | -1.9 | 0.5 | 3.8 |
| -0.5 | 0.6 | 2.8 | 1.7 |
| -0.1 | 1.7 | 4.1 | 1.2 |
| 1.2 | 3.3 | 6.0 | 2.1 |
| 3.1 | 1.6 | 5.8 | 2.0 |
| 1.3 | 1.6 | 4.6 | 0.5 |
| 2.0 | 0.8 | 4.6 | 1.1 |

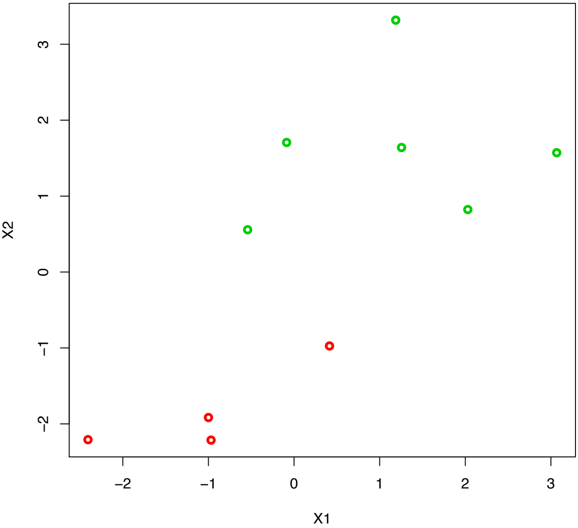
# K-Means: Example

Choose mean with smaller distance:

| $x_1$ | $x_2$ | $(-1.5, -2.1)$ | $(1.0, 1.2)$ |
|---|---|---|---|
| 0.4 | -1.0 | **2.2** | 2.3 |
| -1.0 | -2.2 | **0.5** | 4.0 |
| -2.4 | -2.2 | **1.0** | 4.9 |
| -1.0 | -1.9 | **0.5** | 3.8 |
| -0.5 | 0.6 | 2.8 | **1.7** |
| -0.1 | 1.7 | 4.1 | **1.2** |
| 1.2 | 3.3 | 6.0 | **2.1** |
| 3.1 | 1.6 | 5.8 | **2.0** |
| 1.3 | 1.6 | 4.6 | **0.5** |
| 2.0 | 0.8 | 4.6 | **1.1** |

# K-Means: Example

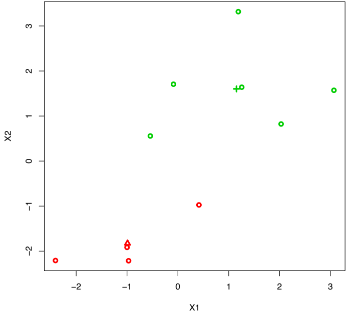
New clusters:



# K-Means: Example

Refit means for each cluster:

- cluster 1: $(0.4, -1.0)$, $(-1.0, -2.2)$, $(-2.4, -2.2)$, $(-1.0, -1.9)$
- new mean: $(-1.0, -1.8)$
- cluster 2: $(-0.5, 0.6)$, $(-0.1, 1.7)$, $(1.2, 3.3)$, $(3.1, 1.6)$, $(1.3, 1.6)$, $(2.0, 0.8)$
- new mean: $(1.2, 1.6)$



# K-Means: Example

Recalculate distances for each cluster:

| $x_1$ | $x_2$ | $(-1.0, -1.8)$ | $(1.2, 1.6)$ |
|---|---|---|---|
| 0.4 | -1.0 | 1.6 | 2.7 |
| -1.0 | -2.2 | 0.4 | 4.4 |
| -2.4 | -2.2 | 1.5 | 5.2 |
| -1.0 | -1.9 | 0.1 | 4.1 |
| -0.5 | 0.6 | 2.4 | 2.0 |
| -0.1 | 1.7 | 3.6 | 1.2 |
| 1.2 | 3.3 | 5.6 | 1.7 |
| 3.1 | 1.6 | 5.3 | 1.9 |
| 1.3 | 1.6 | 4.1 | 0.1 |
| 2.0 | 0.8 | 4.0 | 1.2 |

# K-Means: Example

Select smallest distance and compare these clusters with previous:

Table: New Clusters

| $x_1$ | $x_2$ | $(-1.0, -1.8)$ | $(1.2, 1.6)$ |
|---|---|---|---|
| 0.4 | -1.0 | **1.6** | 2.7 |
| -1.0 | -2.2 | **0.4** | 4.4 |
| -2.4 | -2.2 | **1.5** | 5.2 |
| -1.0 | -1.9 | **0.1** | 4.1 |
| -0.5 | 0.6 | 2.4 | **2.0** |
| -0.1 | 1.7 | 3.6 | **1.2** |
| 1.2 | 3.3 | 5.6 | **1.7** |
| 3.1 | 1.6 | 5.3 | **1.9** |
| 1.3 | 1.6 | 4.1 | **0.1** |
| 2.0 | 0.8 | 4.0 | **1.2** |

Table: Old Clusters

| $(-1.5, -2.1)$ | $(1.0, 1.2)$ |
|---|---|
| **2.2** | 2.3 |
| **0.5** | 4.0 |
| **1.0** | 4.9 |
| **0.5** | 3.8 |
| 2.8 | **1.7** |
| 4.1 | **1.2** |
| 6.0 | **2.1** |
| 5.8 | **2.0** |
| 4.6 | **0.5** |
| 4.6 | **1.1** |

# Outline

# K-Means in R

R has a function for K-means in the `stats` package; this is probably already loaded

- let's use this for the Old Faithful data

```
> library(datasets)
> faith.2 <- kmeans(faithful,2)
> names(faith.2)
> plot(faithful[,1],faithful[,2],col=faith.2$cluster,
 +   pch=faith.2$cluster,lwd=3)
```

# K-Means in R

K-means can be used for *image segmentation*

- partition image into multiple segments
- find boundaries of objects (useful for object recognition)
- make art

# K-Means in R

We can segment memes:

(4 segments on right)

# K-Means in `R`

To segment:

- load image, use `jpeg` package for .jpegs
- coerce input to matrix
- run K-means
- replace existing colors with mean values (or others!) for each cluster
- coerce into array
- save new image

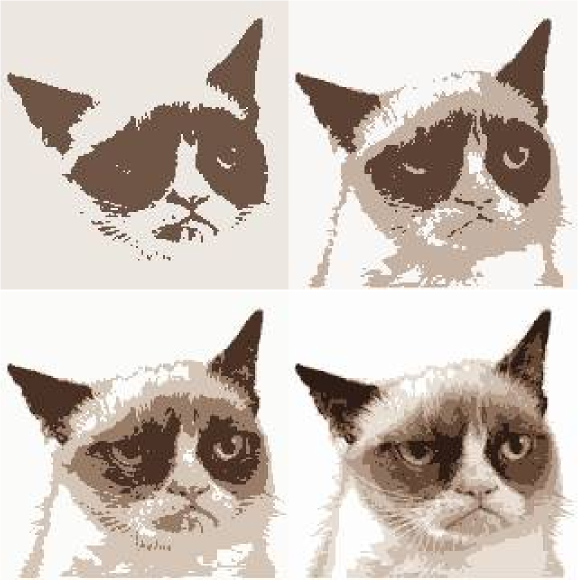
If you want to follow along, load the following packages:

- `jpeg`
- `stats` (probably already loaded)

# K-Means in R

```
> grumpy <- readJPEG("GrumpyTrimmed.jpg")
> source("/DataMining/Lectures/Lecture22/k.means.to.image.R")
> grumpy.4 <- k.means.to.image(grumpy,4)
> writeJPEG(grumpy.4,"GrumpyK4.jpeg")
#==============================
k.means.to.image <- function(im.mat,K){
# image im.mat, number of clusters K # coerce image into matrix
orig.dim <- dim(im.mat)
new.im <- im.mat
dim(new.im) <- c(orig.dim[1]*orig.dim[2],3)
k.list <- kmeans(new.im,K) # Do K means!
out.im <- mat.or.vec(orig.dim[1]*orig.dim[2],3)
for (k in 1:K){
    out.im[(k.list$cluster==k),1] <- k.list$centers[k,1]
    out.im[(k.list$cluster==k),2] <- k.list$centers[k,2]
    out.im[(k.list$cluster==k),3] <- k.list$centers[k,3]}
# Re-coerce new image to original size
dim(out.im) <- orig.dim
return(out.im)
}
```

# K-Means in R



# Outline

# Mixture Models

K-means is similar to the *Gaussian mixture model*

To generate data from a GMM:

- choose cluster with $C_i \sim Categorical(p_1, ..., p_p)$
- generate point $x_i$ with $x_i | C_i = k \sim \mathcal{N}(\mu_k, \Sigma_k)$
- ($\mu_k$ is mean vector, $\Sigma_k$ is covariance matrix)

As with K-Means, we generated data with:

- observation $x_i$ in cluster $C_i$
- $K$ clusters
- Our goal is use data to find $\mu_k$ (and $\Sigma_k$)

# Mixture Models

Mixture models are fit using the following iterative steps:

1. E-step:
$$\mathbb{P}(C_i = k \mid x_i) = \frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_{\ell=1}^{K} \pi_\ell N(x_i \mid \mu_\ell, \Sigma_\ell)}$$
2. M-step:
$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \mathbb{P}(C_i = k \mid x_i) x_i$$
$$\pi_k^{new} = \frac{1}{N} \sum_{i=1}^{n} \mathbb{P}(C_i = k \mid x_i)$$
$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \mathbb{P}(C_i = k \mid x_i)(x_i - \mu_k^{new})(x_i - \mu_k^{new})^\top$$

- (the Expectation-Maximization algorithm)
- Let's compare this conceptually to K-Means

# Mixture Models

Mixture models are closely related to:

- classification with linear/quadratic discriminate analysis

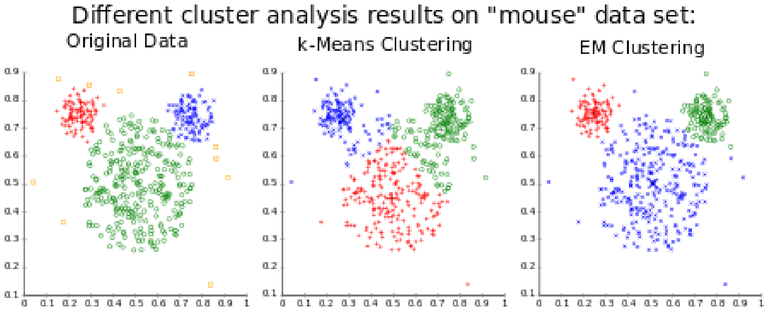
$$\mathbb{P}(y_i = k \mid x_i) = \frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_{\ell=1}^{K} \pi_\ell N(x_i \mid \mu_\ell, \Sigma_\ell)}$$

- naive Bayes (if $\Sigma_k$ is a diagonal matrix)

$$\mathbb{P}(y_i = k \mid x_i) = \frac{\pi_k \prod_{j=1}^{p} N(x_{ij} \mid \mu_{kj}, \sigma_{kj}^2)}{\sum_{\ell=1}^{K} \pi_\ell \prod_{j=1}^{p} N(x_{ij} \mid \mu_{\ell j}, \sigma_{\ell j}^2)}$$

# Mixture Models

Mixture models are more flexible than K-means: each component has $\Sigma_k$ along with $\mu_k$ (how does this relate to LDA/QDA?)




Here, a mixture model is called "EM" after the algorithm used to fit the parameters[2]

[2] Photo credit: Wikipedia

# Outline

# Mixture Models vs K-Means

K-means is quite similar to a mixture model. Fit a simple mixture model, where

$$x_i \,|\, C_i = k \sim N(\mu_k, \Sigma_k)$$

Simplify the covariance model:

- covariance matrices have only diagonal elements,

$$\Sigma_k = \begin{bmatrix} \sigma_{k1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{k2}^2 & \dots & 0 \\ \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & \sigma_{kp}^2 \end{bmatrix}$$

- set $\sigma_{k1}^2 = \cdots = \sigma_{kp}^2$, suppose known and *the same for all components*

# Mixture Models vs K-Means

This simplified mixture model then proceeds as:

- start with random cluster centers
- associate observations to clusters by (log-)likelihood,

$$\begin{aligned}
\ell(x_i \mid C_i = k) &= -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log\left(\prod_{j=1}^{p}\sigma_{kj}^2\right) - \frac{1}{2}\sum_{j=1}^{d}(x_{ij}-\mu_{kj})^2/\sigma_{kj}^2 \\
&\propto -p\log(\sigma_k) - \frac{1}{2\sigma_k^2}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2 \\
&\propto -\sum_{j=1}^{p}(x_{i,j}-\mu_{kj})^2
\end{aligned}$$

- refit centers $\mu_1, \ldots, \mu_K$ given clusters...
- recluster observations...
- stop when no change in clusters

# Mixture Model vs K-Means

Compare mixture model with global variance to K-means:

- clustering with K-means: minimize distance

$$d(x_i, \mu_k) = \sqrt{\sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2}$$

- clustering with single variance mixture model: maximize likelihood

$$\ell(x_i \,|\, C_i = k) \propto -\sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2$$

- update means with K-means: use average

$$\mu_{kj} = \frac{1}{n_k} \sum_{C_i = k} x_{ij}$$

- update means in mixture model: use weighted average

$$\mu_{kj} = \frac{1}{N_k} \sum_{i=1}^{n} \mathbb{P}(C_i = k \,|\, x_i) x_{ij}$$

# Outline

# Fitting Clusters

So what about $K$? How do we find the right value?

- in general, there is no one accepted method
- some people choose it arbitrarily

To figure out evaluation methods, let's look at what happens when $K$ increases

- all clustering models have an objective function (what is this for K-means? Mixture models?)
- when we add a cluster, the value in that objective function decreases (for 'minimize' objective functions or increases for 'maximize' objective functions)
- want to trade off number of clusters against objective function values

# Fitting Clusters

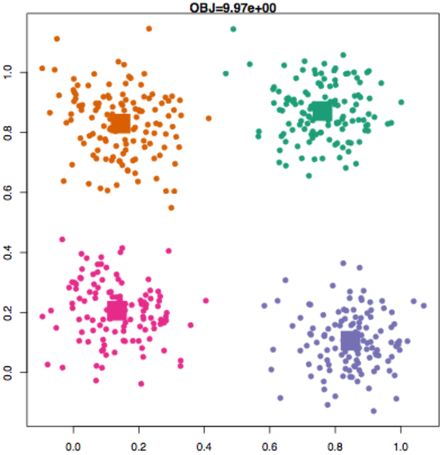



Figure 1: Division of data into four clusters

Credit: Dave Blei

# Fitting Clusters

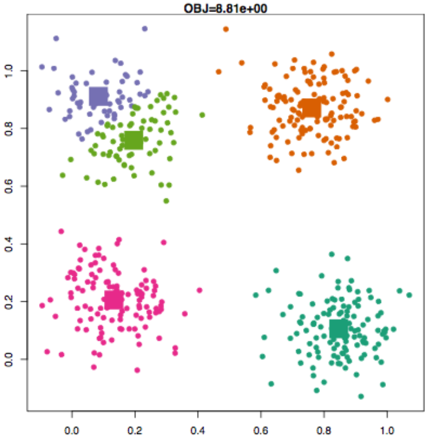



Figure 2: Division of data into five clusters

Credit: Dave Blei

# Fitting Clusters

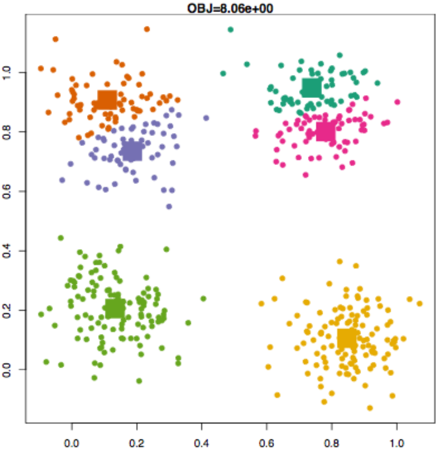



Figure 3: Division of data into six clusters

Credit: Dave Blei

# Fitting Clusters

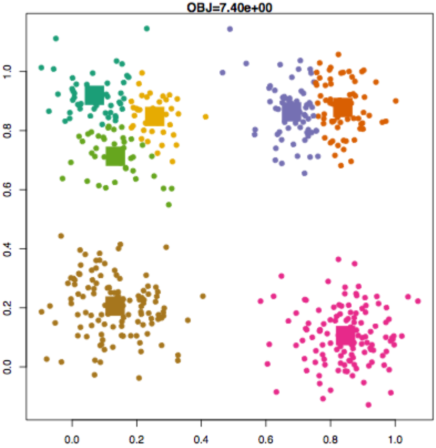



Figure 4: Division of data into seven clusters

Credit: Dave Blei

# Fitting Clusters

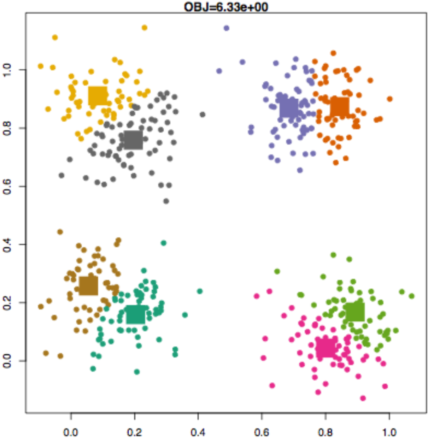



Figure 5: Division of data into eight clusters

Credit: Dave Blei

# Fitting Clusters

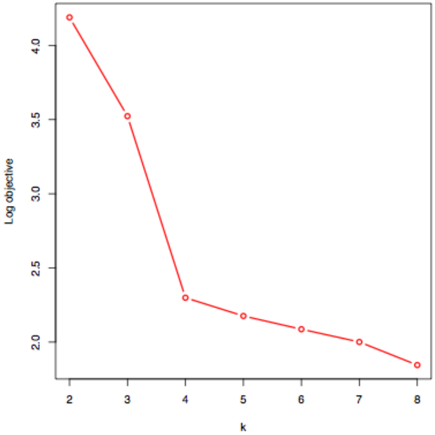



Figure 6: Plot of Log Objective function Vs. number of clusters

Credit: Dave Blei

# Fitting Clusters

Other methods for choosing $K$:

- Akaike Information Criterion (AIC):

$$AIC(K) = 2D_K - 2\log(L)$$

- Bayesian Information Criterion (BIC):

$$BIC(K) = D_K \log(n) - 2\log(L)$$

- in both cases, $L$ is the maximized value of the likelihood function for the model
- $D_K$ is the number of parameters to be estimated with $K$ clusters
- these work for clustering models with likelihoods

# Practice for Final Exam

Use K-means to cluster the following data for $K = 1, 2, 3$ with random initialization:

| $x_1$ | $x_2$ |
|---|---|
| -1.0 | -1.2 |
| -1.2 | -1.8 |
| -2.1 | -2.4 |
| 1.1 | 1.5 |
| 1.5 | 1.6 |
| 1.3 | 0.7 |

# Outline

# Administrative

- **Last 4 office hours**
  - TA: Tuesday Dec $1^{\text{st}}$ and Thursday Dec $3^{\text{rd}}$, 8-9am in 903SSW
  - Prof: Monday Dec 7, 5:30-6pm in 501 Schermerhorn
  - Prof: Wednesday Dec 9, 6-7:30pm in 501 Schermerhorn
- **HW6**
  - will NOT cover SVM
  - is due on Wednesday December 9 <u>BEFORE</u> 6:10pm

# Final Exam

- Date: Wednesday December 16, 2015
- Time: 6 to 8:30pm
- Location: Here (501 Schermerhorn)
- Closed-book: TRUE
- Calculator: Good idea
- Electronic device (smart phone, tablet, laptop): strictly Forbidden