Fuzzy Clustering

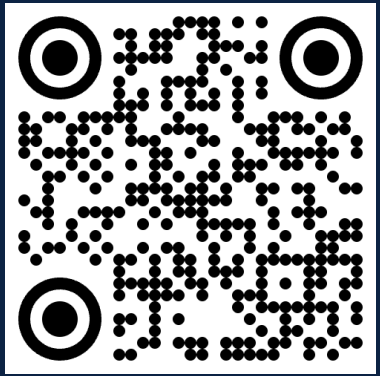
The Advanced ℝ Series: Clustering & Classification

Slides and code created by Jesse Ghashti

January 28, 2026

Centre for Scholarly Communication
The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

New Here?

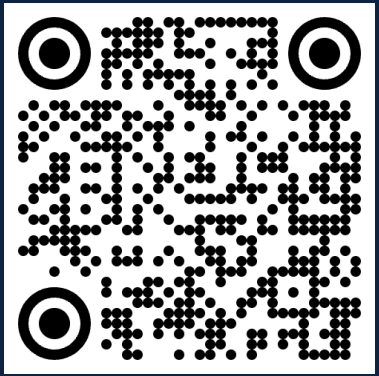


<——————-

Check out our other CSC workshops!

GitHub code and slides for today's workshop (and previous workshops)

——————>



Alternatively, code/slides available at the bottom of `https://csc-ubc-okanagan.github.io/workshops/`

Workshop Series Overview

Session	Topic	Date/Time
1	Hierarchical Models	Jan 14, 12:00 PM
2	Centroid-Based Models	Jan 19, 3:30 PM
3	**Fuzzy Clustering**	**Jan 28, 1:00 PM**
4	Distribution-Based Models	Feb 2, 3:30 PM
5	Density-Based Models	Feb 9, 3:30 PM
6	Graph-Based Models	Feb 23, 3:30 PM
7	Mixed-Type Data Quantification	Mar 4, 1:00 PM
8	Mixed-Type Data Clustering	Mar 11, 1:00 PM
9	Bias Reduction and Fairness	Mar 18, 1:00 PM
10	Dimensionality Reduction 1 of 2	Mar 23, 3:30 PM
11	Dimensionality Reduction 2 of 2	Mar 30, 3:30 PM

Last Time: Centroid-Based Clustering

What we learned...

- Learned K-means algorithm and iterative convergence
- Visualized step-by-step updates and WCSS tracking
- Applied to customer segmentation with data transformation
- Introduced K-medoids (PAM) with Euclidean vs Manhattan distance
- Used elbow method, silhouette, and stability analysis
- Compared classification metrics: Precision, Recall, F1, ARI, NMI

Today: Fuzzy C-means Clustering

Today we will...

- Understand the limitations of hard clustering
- Learn fuzzy C-means (FCM) theory and soft memberships
- Explore the effect of the fuzzy parameter (m)
- Introduce fuzzy-specific validity metrics
- Apply FCM to image segmentation
- Compare hard vs soft clustering approaches

Today we require...

```r
library('e1071')       # fuzzy $C$-means implementation
library('cluster')     # clustering algorithms
library('ggplot2')     # visualization
library('dplyr')       # data manipulation
library('gridExtra')   # plot arrangement
library('imager')      # image processing
```

The Problem with Hard Clustering

K-means forces binary decisions

Hard clustering assumption: Each point belongs to exactly one cluster, but in reality...

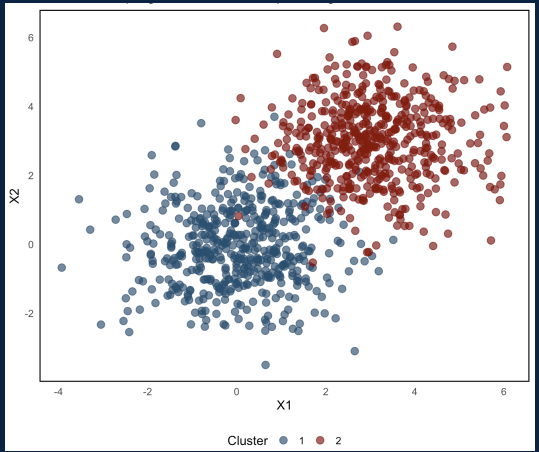
- Overlap regions create ambiguous boundaries
- Points may be equidistant from multiple centroids
- Gradual transitions between clusters are natural
- Binary assignments lose information about uncertainty

Examples include medical diagnosis, image segmentation, customer behaviour

What if we allow **partial memberships** instead of forcing 0/1 assignments?

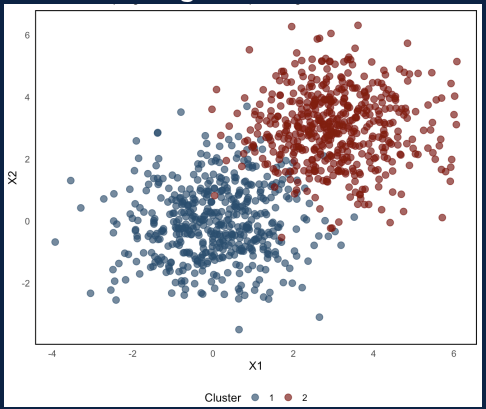
Back to Last Week

```r
set.seed(2025)
nObs <- 1000

cluster1X <- rnorm(nObs/2, mean = 0, sd = 1.2)
cluster1Y <- rnorm(nObs/2, mean = 0, sd = 1.2)
cluster2X <- rnorm(nObs/2, mean = 3, sd = 1.2)
cluster2Y <- rnorm(nObs/2, mean = 3, sd = 1.2)

syntheticData <- data.frame(
  x = c(cluster1X, cluster2X),
  y = c(cluster1Y, cluster2Y),
  trueCluster = factor(rep(c(1, 2), each = nObs/2))
)

ggplot(syntheticData, aes(x = x, y = y,
                          color = trueCluster)) +
  geom_point(alpha = 0.7, size = 2)
  theme_minimal()
```

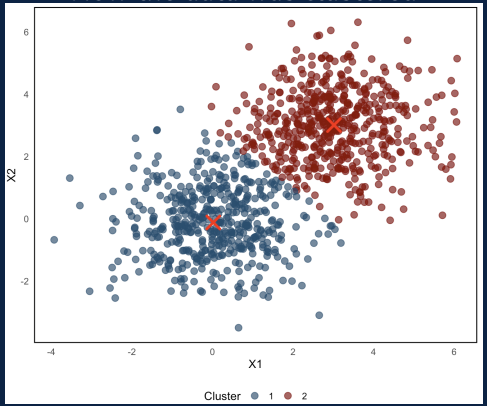


K-means Clustering Results

How we generated the data

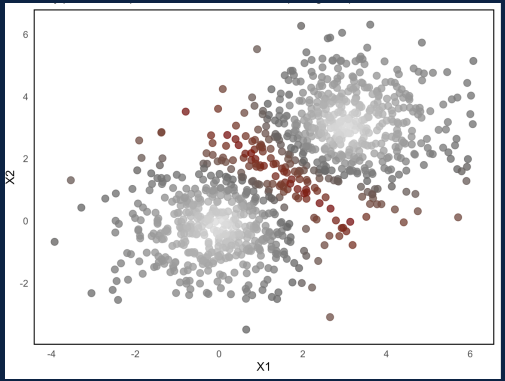


How the data was clustered



Overlapping Regions

```r
kmeansResult <- kmeans(syntheticData[, c("x", "y")],
                       centers = 2, nstart = 25)

syntheticData$kmeansCluster <- factor(kmeansResult$cluster)
kmeansCentroids <- as.data.frame(kmeansResult$centers)
colnames(kmeansCentroids) <- c("x", "y")

distToCentroid1 <- sqrt(
  (syntheticData$x - kmeansCentroids$x[1])^2 +
  (syntheticData$y - kmeansCentroids$y[1])^2)
distToCentroid2 <- sqrt(
  (syntheticData$x - kmeansCentroids$x[2])^2 +
  (syntheticData$y - kmeansCentroids$y[2])^2)

syntheticData$distRatio <-
  pmin(distToCentroid1, distToCentroid2) /
  pmax(distToCentroid1, distToCentroid2)
```



Red points are nearly the same distance from both centroids but forced into one cluster. Should we be confident in the cluster they were assigned?

Fuzzy C-means Objective

Soft Membership Approach

Membership Matrix U: has entries $u_{ic} \in [0, 1]$ for point i in cluster c
Constraint: $\sum_{c=1}^{C} u_{ic} = 1$ for each point i.
Objective Function

$$J_m = \sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^m ||x_i - v_c||^2$$

- v_c = centroid (mean) of cluster c
- m = fuzzy parameter ($m > 1$)
- Higher u_{ic}^m weights points closer to centroid

Fuzzy C-means Algorithm

Iterative Updates (similar to K-means)

Step 1: Update Memberships

$$u_{ic} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{ic}}{d_{ij}} \right)^{\frac{2}{m-1}}}$$

where $d_{ic} = ||x_i - v_c||$ is distance from point i to centroid c

Step 2: Update Centroids

$$v_c = \frac{\sum_{i=1}^{n} u_{ic}^m x_i}{\sum_{i=1}^{n} u_{ic}^m}$$

Repeat until memberships stabilize (they don't change)

Memberships are inversely related to distance, weighted by fuzzifier m

The fuzzy parameter (m)

Controlling Fuzziness

Effect of m on memberships:

- $m \to 1$ approaches hard clustering (last week... o/1 memberships)
- $m = 2$ is the common default (moderate fuzziness)
- $m \to \infty$ means all memberships approach $1/C$ (maximum fuzziness with uniform cluster membership)

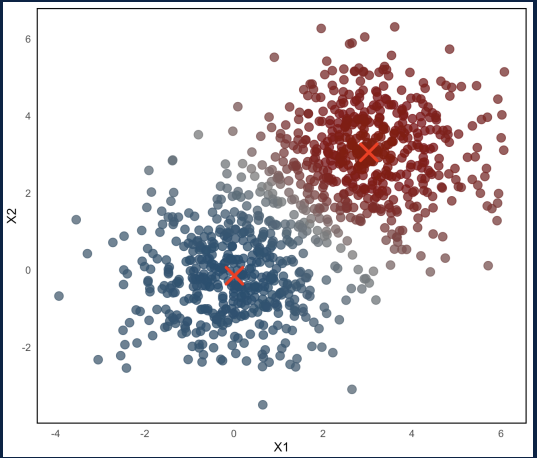
Typical range is $m \in [1.5, 3.0]$

How to choose?

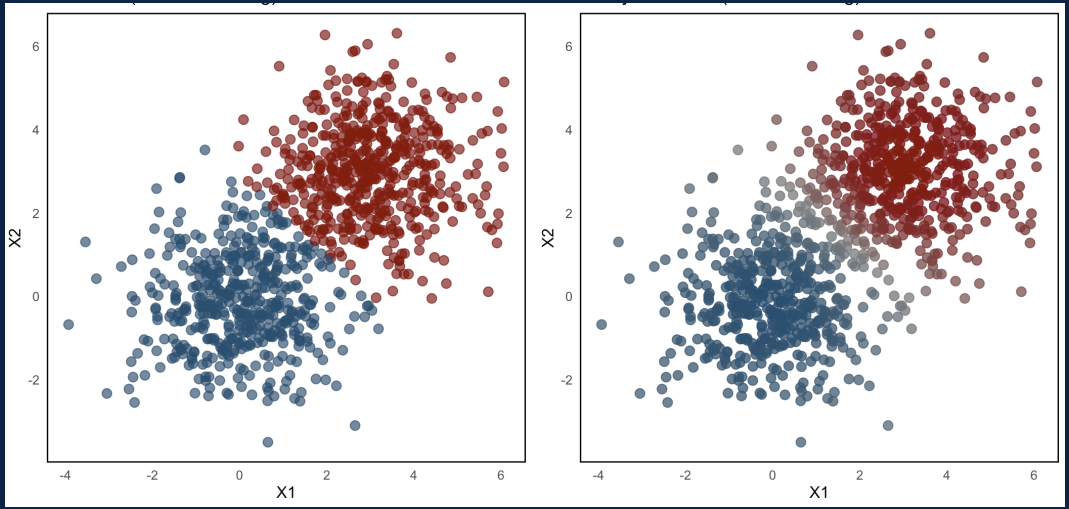
- Domain knowledge about expected overlap
- Cross-validation with fuzzy validity indices
- Visual inspection of membership distributions

Applying Fuzzy C-means

```r
library(e1071)
fcmResult <- cmeans(syntheticData[, c("x", "y")],
                    centers = 2,
                    m = 2,
                    iter.max = 100,
                    verbose = FALSE,
                    method = "cmeans")

print(fcmResult$centers)
print(dim(fcmResult$membership))
print(head(fcmResult$membership))

syntheticData$fcmMembership1 <-
  fcmResult$membership[, 1]
syntheticData$fcmMembership2 <-
  fcmResult$membership[, 2]
syntheticData$fcmCluster <-
  factor(fcmResult$cluster)
```

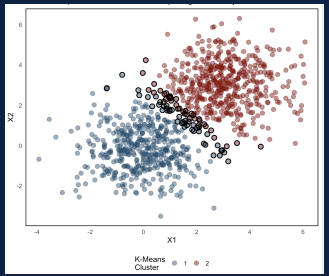


Visualizing Fuzzy Memberships



Those with Approximately Uniform Cluster Membership

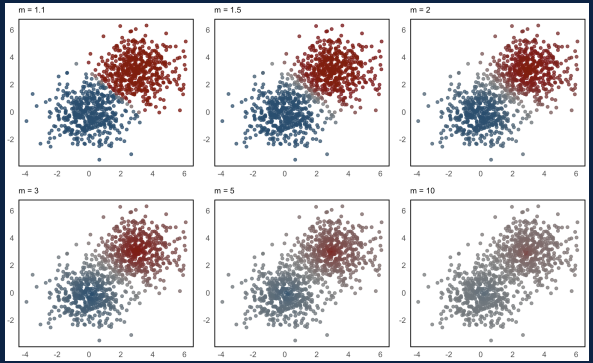
For each point i, if we use $\max_c U_{ic}$ to assign the fuzzy memberships as hard memberships, we should get the K-means results.



	x	y	trueCluster	kmeansCluster	fcmMembership1	fcmMembership2	maxMembership
144	2.0920196	0.91318330	1	2	0.5000428	0.4999572	0.5000428
36	0.4675146	2.43340815	1	1	0.4976978	0.5023022	0.5023022
600	0.7652818	2.17318680	2	2	0.5027915	0.4972085	0.5027915
554	2.6625796	0.39060476	2	2	0.5033129	0.4966871	0.5033129
957	3.1330792	-0.03340036	2	2	0.5056867	0.4943133	0.5056867
515	0.1949789	2.75873355	2	2	0.5113091	0.4886909	0.5113091
184	-0.7978259	3.51103017	1	1	0.4862284	0.5137716	0.5137716
401	2.4005214	0.56650730	1	1	0.4860086	0.5139914	0.5139914
680	2.0781364	1.00143481	2	2	0.5225199	0.4774801	0.5225199
353	0.8675079	1.98552264	1	1	0.4766337	0.5233663	0.5233663

Effect of fuzzy parameter (m)

```
mValues <- c(1.1, 1.5, 2, 3, 5, 10); fuzzifierResults <- list()
for (mVal in mValues) {
  fcmTemp <- cmeans(syntheticData[, c("x", "y")], centers = 2, m = mVal, iter.max = 100, verbose = FALSE)
  membershipEntropy <- -rowSums(fcmTemp$membership * log(fcmTemp$membership + 1e-10))
  fuzzifierResults[[as.character(mVal)]] <- list(m = mVal, membership = fcmTemp$membership, meanMaxMembership = mean(apply(
      fcmTemp$membership, 1, max)), meanEntropy = mean(membershipEntropy))}
```



Fuzzy Clustering Validity Metrics

Metrics for Fuzzy Clustering

Partition Coefficient (PC): $PC = \frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^2$

- Range: $[1/C, 1]$, higher is better
- Measures 'crispness' of partition

Partition Entropy (PE): $PE = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic} \log(u_{ic})$

- Range: $[0, \log(C)]$, lower is better
- Measures fuzziness/uncertainty

Xie-Beni Index: $XB = \frac{\sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^m ||x_i - v_c||^2}{n \cdot \min_{j \neq k} ||v_j - v_k||^2}$

- Lower is better (compactness vs separation)

Computing Fuzzy Validity Metrics

```r
computePartitionCoefficient <- function(membershipMatrix) {
  n <- nrow(membershipMatrix)
  return(sum(membershipMatrix^2)/n)
}
computePartitionEntropy <- function(membershipMatrix) {
  n <- nrow(membershipMatrix)
  membershipSafe <- pmax(membershipMatrix, 1e-10)
  return(sum(membershipMatrix*log(membershipSafe))/n)
}
computeXieBeni <- function(data, membershipMatrix,
                           centers, m = 2) {
  n <- nrow(data); c <- nrow(centers)
  numerator <- 0
  for (i in 1:n) {
    for (j in 1:c) {
      dist2 <- sum((data[i, ] - centers[j, ])^2)
      numerator <- numerator + (membershipMatrix[i, j]^m) * dist2
    }
  }
  minInterCentroid <- Inf
  for (j1 in 1:(c-1)) {
    for (j2 in (j1+1):c) {
      dist2 <- sum((centers[j1, ] - centers[j2, ])^2)
      minInterCentroid <- min(minInterCentroid, dist2)
    }
  }
  return(numerator/(n * minInterCentroid))}
```

k	PC	NPC	PE	NPE	XB
2	0.817	0.635	0.307	0.443	0.110
3	0.655	0.483	0.605	0.551	0.239
4	0.568	0.425	0.794	0.573	0.390
5	0.529	0.412	0.924	0.574	0.289
6	0.499	0.399	1.027	0.573	0.182
7	0.476	0.389	1.116	0.574	0.180
8	0.457	0.379	1.193	0.574	0.176

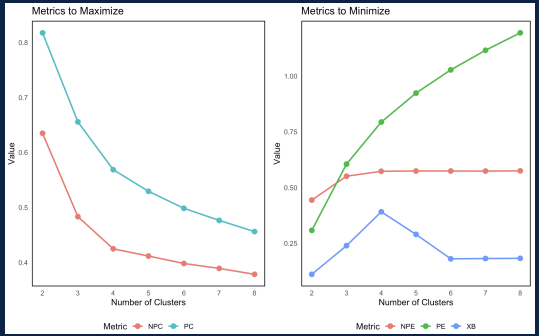
-> PC closer to 1 = less fuzzy

-> PE closer to 0 = less uncertain

-> Lower XB = better separation

Selecting Optimal Number of Clusters

```r
clusterRange <- 2:8
clusterMetrics <- data.frame(
  k = clusterRange,
  PC = numeric(length(clusterRange)),
  PE = numeric(length(clusterRange)),
  XB = numeric(length(clusterRange))
)
for (i in seq_along(clusterRange)) {
  k <- clusterRange[i]
  fcmTemp <- cmeans(dataMatrix, centers = k,
                    m = 2, iter.max = 100,
                    verbose = FALSE)

  clusterMetrics$PC[i] <-
    computePartitionCoefficient(fcmTemp$membership)
  clusterMetrics$PE[i] <-
    computePartitionEntropy(fcmTemp$membership)
  clusterMetrics$XB[i] <-
    computeXieBeni(dataMatrix,
                   fcmTemp$membership,
                   fcmTemp$centers, m = 2)
}
print(clusterMetrics)
```



Fuzzy Image Segmentation

Why FCM for Images?

Image segmentation challenges

- Gradual transitions between objects
- Noise and illumination effects
- Partial volume effects (mixed pixels)
- Shadows and reflections create ambiguous regions

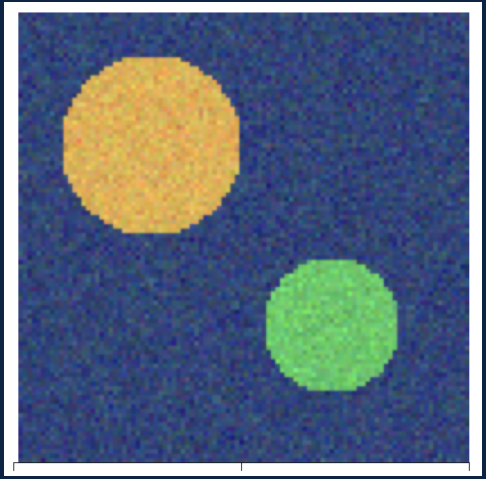
FCM advantages

- Natural handling of boundary pixels
- Uncertainty maps reveal ambiguous regions
- Robust to noise compared to hard segmentation
- Preserves gradual transitions

Each pixel becomes an observation; RGB values become features

Image Segmentation Setup

```
library(imager)
demoWidth <- 100
demoHeight <- 100
demoR <- matrix(0.2, nrow = demoWidth, ncol = demoHeight)
demoG <- matrix(0.3, nrow = demoWidth, ncol = demoHeight)
demoB <- matrix(0.5, nrow = demoWidth, ncol = demoHeight)

for (i in 1:demoWidth) {
  for (j in 1:demoHeight) {
    dist1 <- sqrt((i - 30)^2 + (j - 30)^2)
    dist2 <- sqrt((i - 70)^2 + (j - 70)^2)
    if (dist1 < 20) {
      demoR[i, j] <- 0.9; demoG[i, j] <- 0.7; demoB[i, j] <- 0.3
    }
    if (dist2 < 15) {
      demoR[i, j] <- 0.3; demoG[i, j] <- 0.8; demoB[i, j] <- 0.4
    }
  }
}
demoImg <- array(c(demoR, demoG, demoB),
                 dim = c(demoWidth, demoHeight, 1, 3))
demoImg <- as.cimg(demoImg)
```

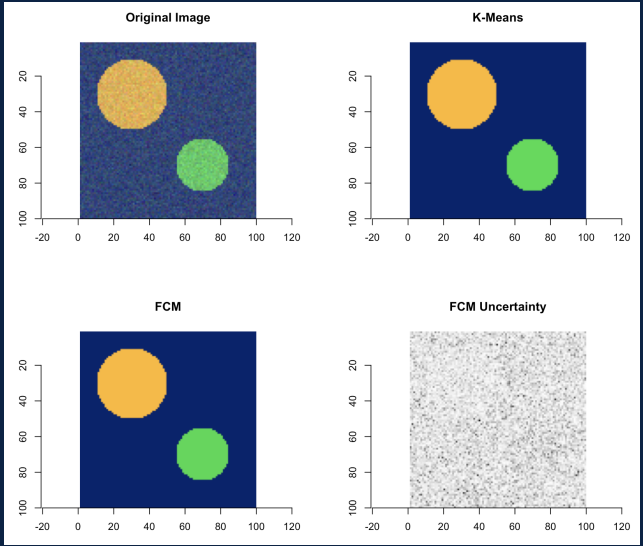


Clustering the Demo Image

```r
demoPixelData <- data.frame(
  x = rep(1:demoWidth, demoHeight),
  y = rep(1:demoHeight, each = demoWidth),
  R = as.vector(demoR),
  G = as.vector(demoG),
  B = as.vector(demoB)
)

demoFeatures <- as.matrix(demoPixelData[, c("R", "G", "B")])

kmeansDemo <- kmeans(demoFeatures, centers = 3, nstart = 25)

fcmDemo <- cmeans(demoFeatures, centers = 3,
                  m = 2, iter.max = 100, verbose = FALSE)

kmeansReconstructed <- reconstructImageHard(
  list(data = demoPixelData, width = demoWidth, height = demoHeight),
  kmeansDemo$cluster,
  kmeansDemo$centers
)

fcmReconstructed <- reconstructImageHard(
  list(data = demoPixelData, width = demoWidth, height = demoHeight),
  fcmDemo$cluster,
  fcmDemo$centers
)
```

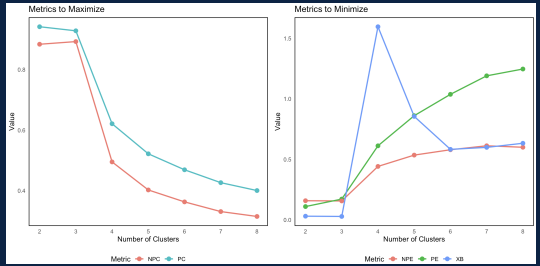
```r
visualizeMembershipUncertainty <- function(
    imgInfo, membershipMatrix) {
  maxMembership <- apply(membershipMatrix, 1, max)
  uncertaintyMap <- 1 - maxMembership
  uncertaintyImg <- array(uncertaintyMap,
    dim = c(imgInfo$width, imgInfo$height, 1, 1))
  return(as.cimg(uncertaintyImg))
}

uncertaintyMap <- visualizeMembershipUncertainty(
  list(data = demoPixelData, width = demoWidth,
       height = demoHeight),
  fcmDemo$membership
)

visualizeClusterMembership <- function(
    imgInfo, membershipMatrix, clusterIdx) {
  clusterMembership <- membershipMatrix[, clusterIdx]
  membershipImg <- array(clusterMembership,
    dim = c(imgInfo$width, imgInfo$height, 1, 1))
  return(as.cimg(membershipImg))
}
```

Membership Uncertainty Maps

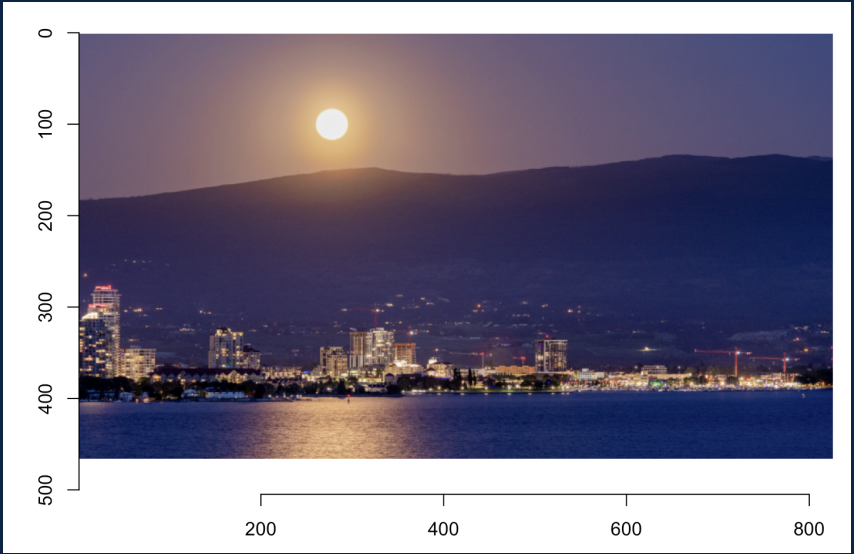


Optimal Clusters for Image Segmentation

```r
imageClusterRange <- 2:8
imageClusterMetrics <- data.frame(
  k = imageClusterRange,
  PC = numeric(length(imageClusterRange)),
  PE = numeric(length(imageClusterRange)),
  XB = numeric(length(imageClusterRange)),
  WithinSS = numeric(length(imageClusterRange))
)

for (i in seq_along(imageClusterRange)) {
  k <- imageClusterRange[i]
  fcmTemp <- cmeans(demoFeatures, centers = k,
                    m = 2, iter.max = 100, verbose = FALSE)
  imageClusterMetrics$PC[i] <-
    computePartitionCoefficient(fcmTemp$membership)
  imageClusterMetrics$PE[i] <-
    computePartitionEntropy(fcmTemp$membership)
  imageClusterMetrics$XB[i] <-
    computeXieBeni(demoFeatures, fcmTemp$membership,
                   fcmTemp$centers, m = 2)
  withinSS <- 0
  for (j in 1:k) {
    clusterPoints <- demoFeatures[
      fcmTemp$cluster == j, , drop = FALSE]
    if (nrow(clusterPoints) > 0) {
      withinSS <- withinSS + sum(sweep(
        clusterPoints, 2, fcmTemp$centers[j, ])^2)
    }}
  imageClusterMetrics$WithinSS[i] <- withinSS}
```

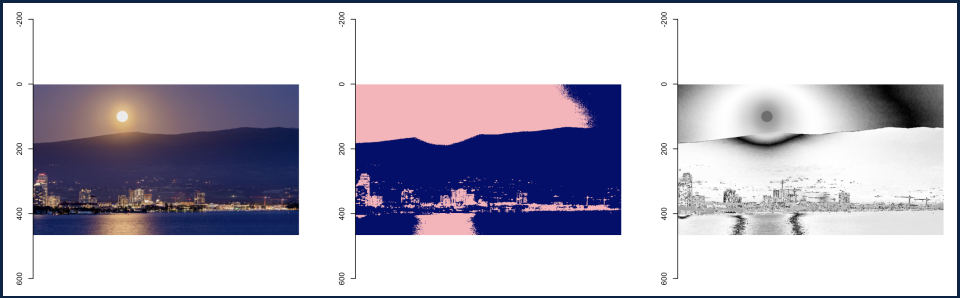


Application to Real Images



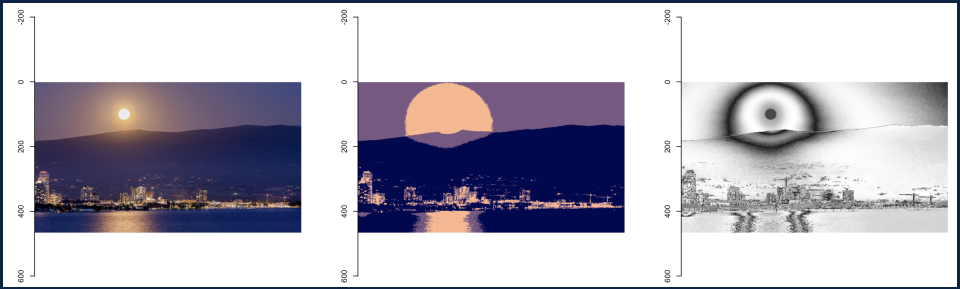
Two Clusters

Best number of clusters suggested by the validity measures



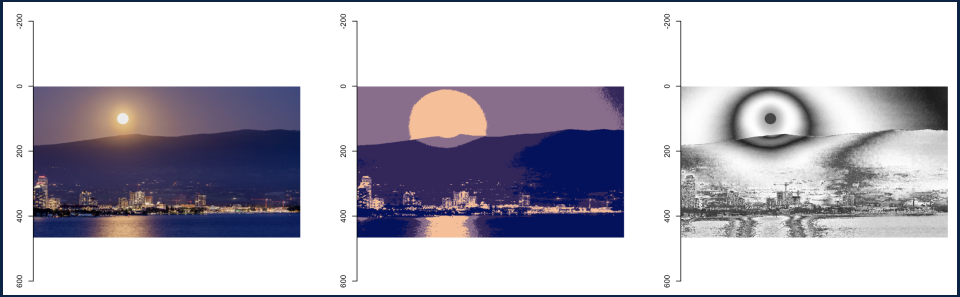
Three Clusters

Second best number of clusters suggested by the validity measures



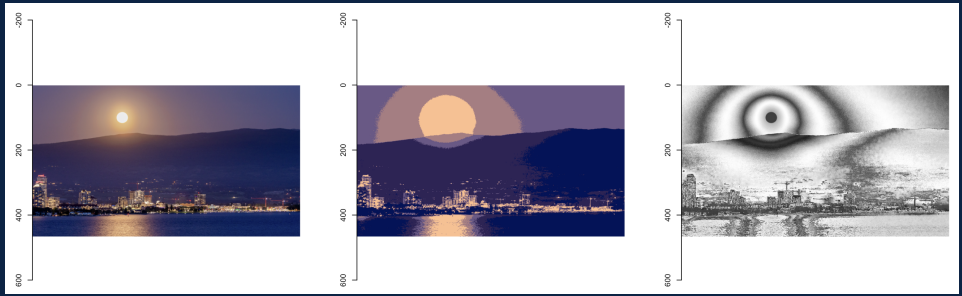
Four Clusters

Just to see...



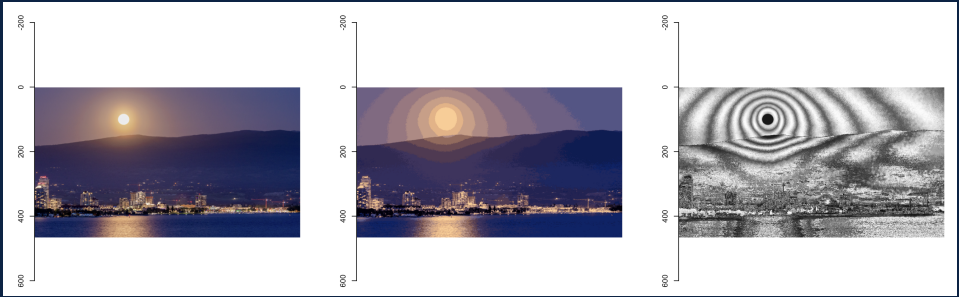
Five Clusters

Just to see...



Twenty Clusters

Just to see...

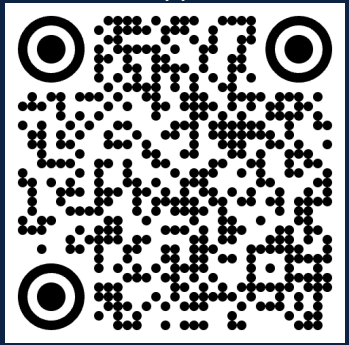


Takeaways

- **Soft clustering** allows partial memberships vs hard 0/1 assignments
- **FCM objective** minimizes weighted sum of squares with membership constraint
- **Fuzzifier m** controls degree of overlap: $m \to 1$ (hard), $m \to \infty$ (maximally fuzzy)
- **Fuzzy metrics** measure partition quality: PC (higher better), PE (lower better), XB (lower better)
- **Membership uncertainty** reveals ambiguous boundaries
- **Image segmentation** benefits from gradual transitions at object boundaries
- **Uncertainty maps** highlight pixels with mixed membership
- **Domain knowledge** helps choose appropriate fuzzifier and cluster count
- **FCM complements** rather than replaces hard clustering
- **Computational cost** higher than K-means due to membership updates

What's Next?

Additional Questions?
Book an Appointment!



Next Workshop: Feb 2, 3:30 PM

Distribution-Based Clustering

-> Workshop 2 (K-means) had circular cluster and hard partition limitations.

-> Workshop 3 (C-means) addressed the hard partition limitations.

-> Next workshop will address the circular cluster limitation.

Thank You!

Questions?

Workshop Materials:
`https://csc-ubc-okanagan.github.io/workshops/`

Contact:
jesse.ghashti@ubc.ca