k-Means Clustering

k-Means Clustering corresponds to a Gaussian classifier with equal diagonal covariance matrix so the decision boundary will be perpendicular to the line connecting the cluster means after the last iteration.

1. Pick k cluster centers using any number of methods, ie. first k points, k equally spaced points.
2. Assign each object to the closest cluster center using Euclidean distance.
3. Recompute the cluster center based on the new labels.
4. Repeat from 2 until number of changes reaches a threshold.

k-Nearest Neighbor

k should be odd and selected a priori. Essentially a method of voting which class an object belongs in based upon Euclidean distance from the sample.

1. Out of N training vectors, identify the k nearest neighbors (measured by Euclidean distance) in the training set, irrespective of the class label
2. Out of these k samples, identify the number of vectors k_i that belong to class ω_i, $i : 1, 2, ..., M$ (if we have M classes)
3. Assign x_i to the class ω_i with the maximum number of k_i samples.

Gaussian classifier

Inherently quadratic. This means that it cannot deal with things like the banana shaped classes

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \qquad (1)$$

Case 1: $\Sigma_i = \sigma^2 I$

The minimum distance classifier with spherical clusters and a decision boundary half way between the cluster means.

$$g_i(x) = -\frac{1}{2}\frac{||x-\mu||^2}{\sigma^2} + \ln P(\omega_i) \qquad (2)$$

Case 2: $\Sigma_i = \Sigma$

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) + \ln P(\omega_i) \tag{3}$$

Watershed

The general principle behind watershed is connecting edges by dilation

Moments

$$m_{p,q} = \sum_x \sum_y x^p y^q f(x,y) \tag{4}$$

Figure 1: An ordinary or raw moment

Central moments are independent of position, but not scaling or rotation invariant.

For an object to exhibit a unique orientation $\mu_{02} \neq \mu_{20}$

$$\mu_{p,q} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y) \tag{5}$$

Figure 2: A position invariant, central moment

Area	m_{00}
Center of mass	m_{01} or m_{10}
Variance	μ_{02} or μ_{20}
Covariance	μ_{11}

$$\theta = \frac{1}{2}\tan^{-1}\frac{2\mu_{11}}{\mu_{20}-\mu_{02}} \tag{6}$$

Figure 3: Object orientation derived from 2. order moments

$$f - (f \circ b) = f - ((f \ominus b) \oplus b) \tag{7}$$

Figure 4: Top-hat, detects like objects on a dark background

$$(f \bullet b) - f = ((f \oplus b) \ominus b) - f \tag{8}$$

Figure 5: Bottom-hat, detects dark objects on a light background

Hats

Gray-level co-occurrence matrix

GLCM texture considers the relation between two pixels at a time, called the reference and the neighbour pixel.

First order texture measures are statistics calculated from the original image values, like variance, and do not consider pixel neighbour relationships.

Second order measures consider the relationship between groups of two (usually neighbouring) pixels in the original image.

Third and higher order textures (considering the relationships among three or more pixels) are theoretically possible but not commonly implemented due to calculation time and interpretation difficulty.

- Reference and neighbor pixel offset
- Strategy for handling borders
- Number of gray-levels G
- Window size
- Weighting function(s) if you want to generate a texture image

Figure 6: GLCM Parameters

Applying a weighting function to the GLCM generates a texture image. The weighting functions can be broken up into at least two main categories.

Dissimilarity and Contrast result in larger numbers for more contrasty windows. If weights decrease away from the diagonal, the result will be larger for windows with little contrast.

1. Initialize a GLCM matrix $G \times G$ in size
2. Remove noise, equalize histogram, requantize the image as desired
3. Move across image recording the relationship between the reference and neighbor pixel in the GLCM matrix.
4. Transpose and add the matrix to itself to make it symmetric
5. Normalize the GLCM by dividing each pixel by the sum of the counts converting the table from simple counts to something approximating a probability table

Figure 7: GLCM Algorithm

Value	ASM (Uniformity), Entropy	Comparing values within a matrix
Position	IDM (Homogeneity), Inertia	Position within the matrix

Figure 8: Weighting functions

Homogeneity weights values by the inverse of the Contrast weight, with weights decreasing exponentially away from the diagonal:

$$IDM = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{P_{i,j}}{1 + (i-j)^2} \tag{9}$$

$$Entropy = -\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} P_{i,j} * \ln(P_{i,j}) \tag{10}$$

ASM and Energy use each $P_{i,j}$ as a weight for itself. High values of ASM or Energy occur when the window is very orderly.

$$ASM = -\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (P_{i,j})^2 \tag{11}$$

Hough

Hough is used to connect points into edges or lines.

In the randomized version we use pairs of randomly chosen pixel pairs and calculate θ directly rather than calculating all ρ values for each pixel. This leads to many fewer increments in the matrix.

1. Establish an accumulator matrix of size ρ by θ where ρ is the range of possible magnitudes $\pm\sqrt{rows^2 + cols^2}$ and θ is all angles $\pm 90°$
2. Clean up image: histogram equalize, etc.
3. Apply a gradient edge detector like Sobel to the image
4. Threshold into a binary image
5. Go through each pixel and if lit use each θ value in the range $\pm 90°$ calculating $\rho = x\cos\theta + y\sin\theta$. For each ρ, θ pair increment the corresponding cell in the accumulator matrix.
6. Any local maxima in the accumulator matrix that crosses a predefined threshold is considered a line.

Figure 9: Hough lines

Feature selection

Forward and backward selection involves optimizing the set of features through a systematic strategy of adding and removing of features.

Region growing

1. Choose starting seeds
2. Decide on a predicate to determine if part of region
3. Continue as long as items can be added