INF 4300
10.10.16

Multivariate classification

Anne Solberg (anne@ifi.uio.no)

Based on Chapter 2 (2.1-2.6) in Duda and Hart: Pattern Classification

Today's focus

- From a d-dimensional feature vector $\mathbf{x}=[x_1,....x_d]^T$
- Given K different classes k=1,...K
- Compute the probability that x belongs to class k $P(k|\mathbf{x})= p(\mathbf{x}|k)P(k)/const$
- How should the multivariate density $p(\mathbf{x}|k)$ be?

Bayes rule for a classification problem

- Suppose we have J, j=1,...J classes. ω is the class label for a pixel, and $\boldsymbol{x}$ is the observed feature vector).
- We can use Bayes rule to find an expression for the class with the highest probability:

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j)P(\omega_j)}{p(\mathbf{x})}$$

$$\text{posterior probability} = \frac{likelihood \times \text{prior probability}}{\text{normalizing factor}}$$

- $P(\omega_j)$ is the prior probability for class ω_j. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).

Bayes rule explained

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j)P(\omega_j)}{p(\mathbf{x})}$$

- $p(\mathbf{x}|\omega_j)$ is the probability density function that models the likelihood for observing feature vector $\mathbf{x}$ if the pixel belongs to class ω_j.
  - Typically we assume a type of distribution, e.g. Gaussian, and the mean and covariance of that distribution is fitted to some data that we know belong to that class. This fitting is called classifier training.
- $P(\omega_j|\mathbf{x})$ is the posterior probability that the pixel actually belongs to class ω_j given the observed feature vector $\mathbf{x}$.
- $p(\mathbf{x})$ is just a scaling factor that assures that the probabilities sum to 1.

The conditional density p(**x**| ω_s)

- Any probability density function can be used to model p(**x**| ω_s)
- A common model is the multivariate Gaussian density.
- The multivariate Gaussian density:

$$p(\mathbf{x} \mid \omega_s) = \frac{1}{(2\pi)^{n/2}\left|\boldsymbol{\Sigma}_s\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_s)^t \boldsymbol{\Sigma}_s^{-1}(\mathbf{x}-\boldsymbol{\mu}_s)\right]$$

- If we have d features, $\boldsymbol{\mu}_s$ is a vector of length d and and $\boldsymbol{\Sigma}_s$ a d×d matrix (depends on class *s*)

$$\boldsymbol{\mu}_S = \begin{bmatrix} \mu_{1s} \\ \mu_{2s} \\ \\ \mu_{ns} \end{bmatrix} \qquad \boldsymbol{\Sigma}_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & . & . & . \\ \sigma_{31} & \sigma_{11} & . & . & . \\ . & . & . & . & . \\ \sigma_{n1} & \sigma_{n2} & . & \sigma_{nn-1} & \sigma_{nn} \end{bmatrix}$$

Symmetric d×d matrix
σ_{ii} is the variance of feature i
σ_{ij} is the covariance between feature i and feature j
Symmetric because $\sigma_{ij} = \sigma_{ji}$

- $|\boldsymbol{\Sigma}_s|$ is the determinant of the matrix $\boldsymbol{\Sigma}_s$, and $\boldsymbol{\Sigma}_s^{-1}$ is the inverse

Mean vectors and covariance matrices in d dimensions

- If **x** is a d-dimensional feature vector for one object/pixel, we can formulate its mean vector and covariance matrix as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \end{bmatrix} \qquad \boldsymbol{\mu} = E[\mathbf{x}] = \begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_d) \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ \mu_d \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_{dd} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & . & . & \sigma_{2d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_d^2 \end{bmatrix}$$

- with d features, the mean vector μ will be of size 1xd and Σ of size dxd.

Inspecting p(**x**|ω_s)

$$p(\mathbf{x} \mid \omega_s) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_s\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_s)^t \boldsymbol{\Sigma}_s^{-1}(\mathbf{x}-\boldsymbol{\mu}_s)\right]$$

Scalar probability

Scalar

1xn vector transposed

nxn matrix
Inverse of covariance matrix

nx1 vector transposed

The mean vectors μ_s for each class

- The mean vector for class s is defined as the expected value of **x**:

$$\boldsymbol{\mu}_s = E[\mathbf{x}] = \begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_d) \end{bmatrix} = \begin{bmatrix} \mu_1^s \\ \mu_2^s \\ . \\ . \\ \mu_d^s \end{bmatrix}$$

class s
feature number d

- with d features, the mean vector μ will be of size 1xd.
- If we have M_s training samples that we know belong to class s, we can estimate the mean vector as:

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s}\sum_{m=1}^{M_s} \mathbf{x}_m,$$

where the sum is over all training samples belonging to class s

Link to moments

- From lecture on moments:

$$m_{10} = \sum_x \sum_y x f(x,y) = \bar{x} m_{00} \quad \Rightarrow \quad \bar{x} = \frac{m_{10}}{m_{00}}$$

$$m_{01} = \sum_x \sum_y y f(x,y) = \bar{y} m_{00} \quad \Rightarrow \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

- If **f**=[x,y] is a sample from distribution p(x,y), the mean is defined as

$$\mu_x = \sum_x \sum_y x p(x,y)$$

$$\mu_y = \sum_x \sum_y y p(x,y)$$

The covariance matrix Σ_s for each class

- The covariance for class s is defined as the expected value of $(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t$:

$$\boldsymbol{\Sigma}_s = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdot & \cdot & \sigma_{d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{d1} & \sigma_{d2} & \cdot & \cdot & \sigma_{dd} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdot & \cdot & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{d1} & \sigma_{d2} & \cdot & \cdot & \sigma_d^2 \end{bmatrix}$$

- with d features, the covariance matrix $\boldsymbol{\Sigma}_s$ will be of size dxd.
- If we have M_s training samples that we know belong to class s, we can estimate the covariance matrix $\boldsymbol{\Sigma}_s$. (The estimate of a random variable f is denoted $\hat{f}$)

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} (\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)^t$$

where the sum is over all training samples belonging to class s

- Each term σ_{ij} is computed as:

$$\sigma_{ij,s}^{\ 2} = \frac{1}{M_s} \sum_{m=1}^{M_s} (x_{m,i} - \hat{\mu}_{i,s})(x_{m,j} - \hat{\mu}_{j,s})^t$$

for the covariance between feature i and j for class s

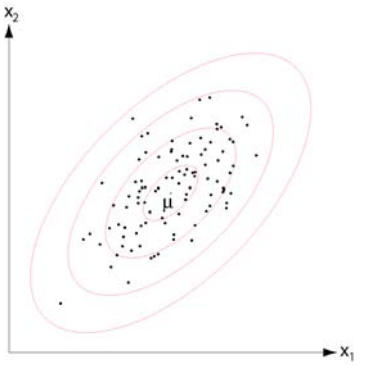
More on the covariance matrix Σ_s

- The covariance matrix Σ_s will always be symmetric and positive semidefinite.
- If all components of x have non-zero variance, Σ_s will be positive definite.
- σ_{ij} is the covariance between features *i* and *j*.
- If features x_i and x_j are uncorrelated, $\sigma_{ij} = 0$.
- In the general case, Σ_s will have d(d+1)/2 different values.

A 2D Gaussian model

- Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ define a density as a "bump"
- The curves on the plot are contours of equal probability, just as the contours on a map
- The matrix $\boldsymbol{\Sigma}$ in this case has three different elements, variance in each of the axes, and covariance between the axes

$$\Sigma_S = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$



- σ_{11}^2 is the variance for feature 1
- $\sigma_{12}=\sigma_{21}$ is the covariance between feature 1 and 2
- σ_{22}^2 is the variance for feature 2

From lecture on moments: Moments of inertia or Variance

- The two second order central moments measure the spread of points around the y- and x-axis through the centre of mass

$$\mu_{20} = \sum_x \sum_y (x-\bar{x})^2 f(x,y)$$

$$\mu_{02} = \sum_x \sum_y (y-\bar{y})^2 f(x,y)$$

- **What does μ_{20} and μ_{02} correspond to in this lecture (assume we have to features)?**

- The cross moment of intertia is given by

$$\mu_{11} = \sum_x \sum_y (x-\bar{x})(y-\bar{y}) f(x,y)$$

  - Statisticians call this covariance or correlation.

- Orientation of the object can be derived from these moments.

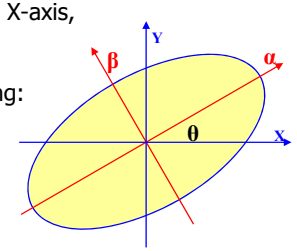
From lecture on moments: Object orientation

- Orientation is defined as the angle, relative to the X-axis, of an axis through the centre of mass that gives the lowest moment of inertia.
- Orientation θ relative to X-axis found by minimizing:

$$I(\theta) = \sum_\alpha \sum_\beta \beta^2 \, f(\alpha,\beta)$$

where the rotated coordinates are given by

$$\alpha = x\cos\theta + y\sin\theta, \quad \beta = -x\sin\theta + y\cos\theta$$



- We found that object orientation was given by:

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{(\mu_{20}-\mu_{02})}\right], \quad \text{where } \theta \in [0, \pi/2] \text{ if } \mu_{11} > 0, \; \theta \in [\pi/2, \pi] \text{ if } \mu_{11} < 0$$

Can we use this to find the orientation of the covariance matrix?

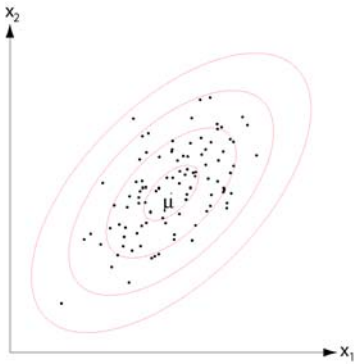
The covariance matrix and ellipses

- In 2D, the Gaussian model can be thought of as approximating the classes in 2D feature space with ellipses.
- The mean vector $\mu=[\mu_1, \mu_2]$ defines the the center point of the ellipses.
- σ_{12}, the covariance between the features defines the orientation of the ellipse.
- σ_{11} and σ_{22} defines the width of the ellipse.

$$\Sigma_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

- The ellipse defines points where the probability density is equal
  - Equal in the sense that the distance to the mean as computed by the Mahalanobis distance is equal.
  - The Mahalanobis distance between a point x and the class center μ is:

$$r^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$

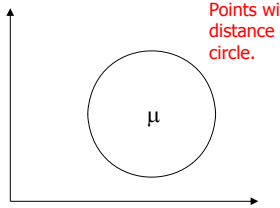


The main axes of the ellipse is determined by the eigenvectors of Σ.
The eigenvalues of Σ gives their length.

Euclidean distance vs. Mahalanobis distance

- Euclidean distance between point x and class center μ:

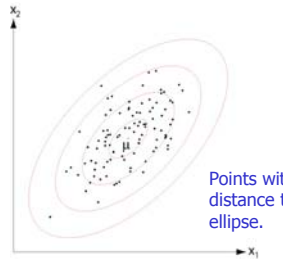
$$(x-\mu)^T (x-\mu) = \|x-\mu\|^2$$



Points with equal distance to μ lie on a circle.

- Mahalanobis distance between x and μ:

$$r^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$



Points with equal distance to μ lie on an ellipse.

Discriminant functions for the normal density

- When finding the class with the highest probability, these functions are equivalent:

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}$$

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i)$$

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$$

- Let us now look at $g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$
- With a multivariate Gaussian we get:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

- Let ut look at this expression for some special cases:

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

- In this case we assume that the features are uncorrelated (independent) with the same variance σ^2
- The covariances $\sigma_{ij}=0$ (by definition if the features are uncorrelated).
- The discriminant functions can be expressed as:

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

$$\text{where } \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = (\mathbf{x}-\boldsymbol{\mu}_i)^t(\mathbf{x}-\boldsymbol{\mu}_i)$$

- Thus we model the probabilities as n-dimensional *spheres* because points that have equal discriminant function will lie on a circle around the mean μ_i.
- $\boldsymbol{\Sigma}_j^{-1} = \mathbf{I}/\sigma^2$
- $|\boldsymbol{\Sigma}_j| = \sigma^{2n}$

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

- The discriminant functions simplifies to **linear** functions using such a shape on the probability distributions

$$g_j(\mathbf{x}) = -\frac{1}{2(\sigma^2 I)}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sigma^2 I| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\mathbf{x} - 2\boldsymbol{\mu}_j^T\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sigma^2 I| + \ln P(\omega_j)$$

Common for all classes, no need to compute these terms
Since $\mathbf{x}^T\mathbf{x}$ is common for all classes, an equivalent $g_j(\mathbf{x})$ is a linear function of $\mathbf{x}$: .

$$\frac{1}{(\sigma^2)}\boldsymbol{\mu}_j^T\mathbf{x} - \frac{1}{2(\sigma^2)}\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j + \ln P(\omega_j)$$

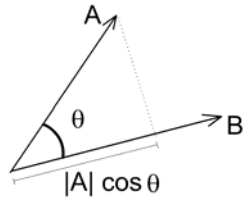
Linear algebra basics: Inner product between two vectors.

- The inner product (or dot product) between two vectors (of length N)a and b or is given by

$$\langle a,b\rangle = \sum_{i=1}^{N} a_i b_i = a^T b$$

- The angle between two vectors A and B is defined as:

$$\cos\theta = \frac{\langle A,B\rangle}{\|A\|\|B\|}$$



- If the inner product of two vectors is zero, they are normal to each other.

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

- Now we get an equivalent formulation of the discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } w_{i0} = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0)=0$$

$$\text{where } \mathbf{w}=\boldsymbol{\mu}_i-\boldsymbol{\mu}_j$$

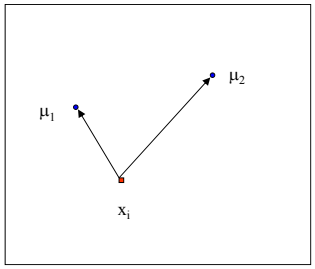
$$\text{and } x_0 = \frac{1}{2}(\boldsymbol{\mu}_i+\boldsymbol{\mu}_j) - \frac{\sigma^2}{\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2}\ln\left[\frac{P(\omega_i)}{P(\omega_j)}\right](\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)$$

- $\mathbf{w}=\boldsymbol{\mu}_i-\boldsymbol{\mu}_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.
- If $P(\omega_i)=P(\omega_j)$ the hyperplane will be located halfway between the mean values.
- Proving this involves some algebra, see the proof at https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

- If the features were indepenent ($\Sigma_j=\sigma^2 I$) the discriminant function was simplified to:

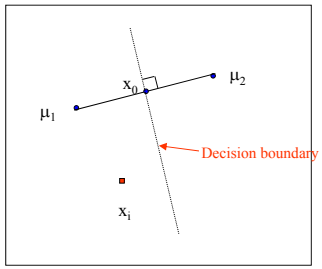
$$g_j(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j)+\ln P(\omega_j)$$

$$= -\frac{1}{2\sigma^2}\|\mathbf{x}-\boldsymbol{\mu}_j\|^2 + \ln P(\omega_j)$$

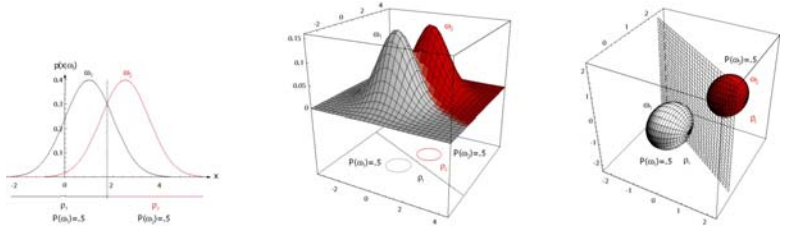


- This results in linear decision boundaries.
- Computing this discriminant function to classify pattern x_i involves computing the distance from the point to the mean values μ_c for each class.

- The discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $x_0=(\mu_1-\mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



A simple model, $\Sigma_j=\sigma^2 I$



- The distributions are spherical in *d* dimensions.
- The decision boundary is a generalized hyperplane of *d-1* dimensions
- The decision boundary is perpendicular to the line separating the two mean values
- This kind of a classifier is called a linear classifier, or a linear discriminant function
  - Because the decision function is a linear function of ***x***.
- If $P(\omega_i)= P(\omega_j)$, the decision boundary will be half-way between μ_i and μ_j

Minimum distance classification

- If all classes have equal prior probabilities, x_0 will be the point halfway between the mean vectors.
- Classification will consist of assigning feature vector x to the same class as the closest mean measured by Euclidean distance $||x-\mu_i||$.
- A classifier based on the Euclidean distance is called a **minimum distance classifier**.

Case 2: Common covariance, $\Sigma_j = \Sigma$

- If we assume that all classes have the same shape of data clusters, an intuitive model is to assume that their probability distributions have the same shape
- By this assumption we can use all the data to estimate the covariance matrix
- This estimate is common for all classes, and this means that also in this case the discriminant functions become linear functions

$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

Common for all classes, no need to compute

Since $\boldsymbol{x}^T\boldsymbol{x}$ is common for all classes, $g_j(\boldsymbol{x})$ again reduces to a linear function of $\boldsymbol{x}$.

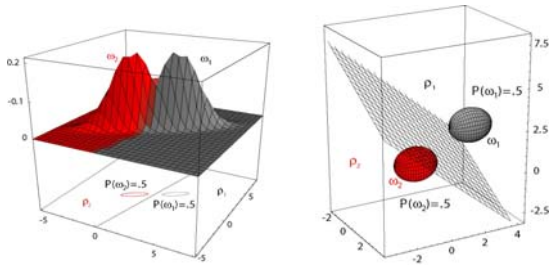
Case 2: Common covariance, $\Sigma_j = \Sigma$

- An equivalent formulation of the discriminant functions is

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + wi_0$$
$$\text{where } \mathbf{w}_i = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i$$
$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- The decision boundaries are again hyperplanes.
- Because $\mathbf{w}_i = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ is not in the direction of $(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$, the hyperplan will not be orthogonal to the line between the means.

Common covariance, $\Sigma_j = \Sigma$



- The classes can be described by hyperellipsoides in *d* dimensions.
- All hyperellipsoids have the same orientation.
- The decision boundary will again be a hyperplane.
- Because $\boldsymbol{w} = \Sigma^{-1}(\mu_i - \mu_j)$ is generally not in the direction of $\mu_i - \mu_j$, the hyperplane will not be perpendicular to the line between the means.
- Consider a point x_0 on the line $\mu_i - \mu_j$, defined by the prior probabilities:
  - If $P(\omega_i) = P(\omega_j)$, x_0 will be half way between the means.
  - The separating hyperplane will *intersect* the line at x_0

Case 3:, Σ_j=arbitrary

- When all classes are modeled as having different *shapes,* the discriminant functions cannot be simplified
- This means that the discriminant functions will be *quadratic* functions

- Decision boundaries will be hyperquadrics and assume any of the general forms:
  - hyperplanes, pairs of hyperplanes, hyperspheres, hyperellisoides, hyperparaboloids, hyperhyperboloids...

Case 3:, Σ_j=arbitrary

- The discriminant functions will be quadratic:

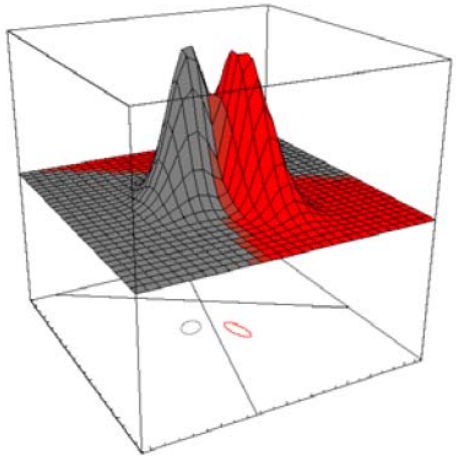
$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + wi_0$$

$$\text{where } \mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \quad \mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

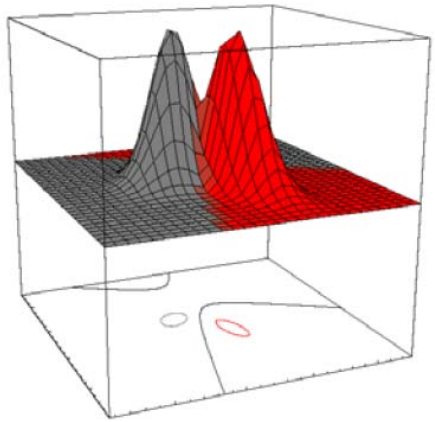
$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i^t \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$

- The decision surfaces are hyperquadrics and can assume any of the general forms:
  - hyperplanes
  - hypershperes
  - pairs of hyperplanes
  - hyperellisoids,
  - Hyperparaboloids,..
- The next slides show examples of this.
- In this general case we cannot intuitively draw the decision boundaries just by looking at the mean and covariance.

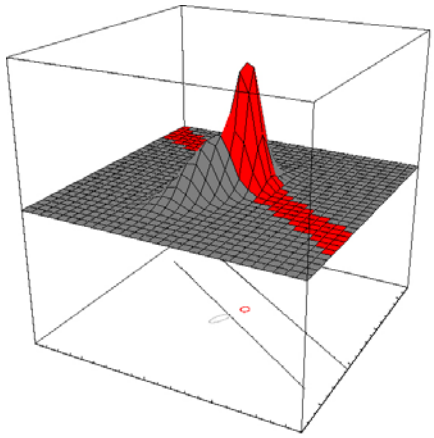
The full model, Σ_j=arbitrary - example



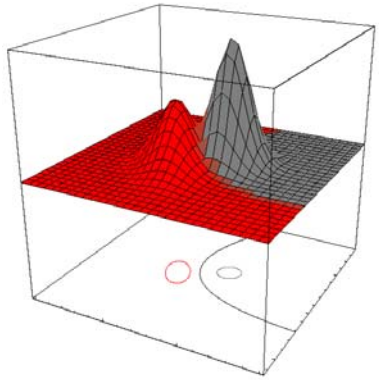
The full model, Σj=arbitrary - example



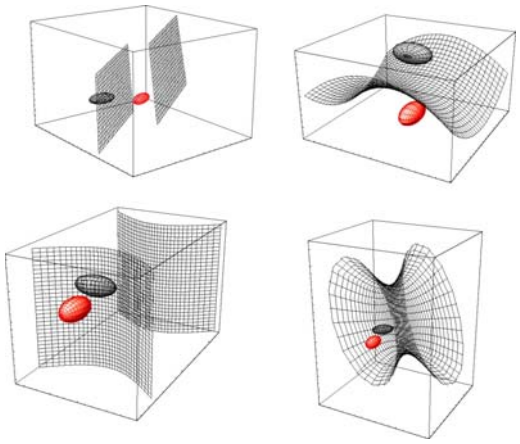
The full model, Σ_j=arbitrary - example



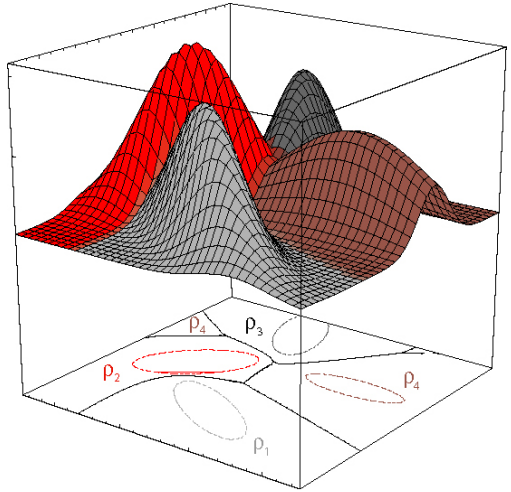
The full model, Σj=arbitrary - example



The full model, Σj=arbitrary - example



A multiclass example

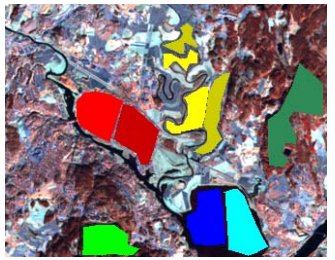


Is the Gaussian classifier the only choice?

- The Gaussian classifier gives linear or quadratic discriminant function.
- Other classifiers can give arbitrary complex decision surfaces (often piecewise-linear)
  - Mixtures of Gaussians
  - Other probability density functions (t-distribution, exponetial distributions).
  - Neural networks
  - Support vector machines
  - Ensembles of simple classifiers
    ADAboost
    Random forest/decision trees
  - kNN (k-Nearest-Neighbor) classification

Using masks to train and test

• Training mask: a mask where regions to train each class are marked using different pixel values, e.g. class label=1 for class 1, 2 for class 2 etc.
•Test mask: a similar mask as training, but to estimate classifier accuracy only.



Training a classifier

- Obtain as many ground truth samples for each class as possible
  - If visual inspection is reliable, experts can mark training regions interactively.
  - For remote sensing, go out in the field and collect field samples (or use images from a different sensor)
  - For symbol recognition, mark a set of symbols manually.
  - For medical applications, use e.g. tissue samples or interpretations made by experts.
- Divide the ground truth into a training set and a test set.
- Use feature extraction and feature selection/evaluation to determine the best set of features.

- Decide if a linear or quadratic classifier is needed.

$\hat{\mu}_s$ has n elements

$\hat{\Sigma}_s$ has n(n-1)/2 elements

Estimating μ_s and Σ_s

- For each class, compute μ_s (and Σ_s) either with a for-loop on each feature, or use a vector implementation.

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s}\sum_{m=1}^{M_s}\mathbf{x}_m,$$

where the sum is over all training samples belonging to class s

$$\hat{\Sigma}_s = \frac{1}{M_s}\sum_{m=1}^{M_s}(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)^t$$

where the sum is over all training samples belonging to class s

$$\sigma_{ij,s}{}^2 = \frac{1}{M_s}\sum_{m=1}^{M_s}(x_{m,i} - \hat{\mu}_{i,s})(x_{m,j} - \hat{\mu}_{j,s})^t$$

for the covariance between feature i and j for class s

Training

```
for i=1:N
  for j=i:M
    if mask(i,j)>==K
      increment nof. Samples in class K
      store the feature vector f(i,j) in a vector of training samples from class K
    end
end
end

For class k=1:K
    compute mean(k) and sigma(k)
```


Classifying new data

- For each sample, compute the posterior probabilities for each class.

$$P(\omega_s \mid x) \propto p(x \mid \omega_s) P(\omega_s)$$

$$= \left[\frac{1}{(2\pi)^{P/2} \left|\Sigma_s\right|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_s)^t \Sigma_s^{-1} (x - \mu_s) \right] \right] P(\omega_s)$$

- Classify the sample to the class with the highest posterior probability.
- Evaluate the performance of the classifier on a different dataset.
- We can also produce images of the posterior probability for each class.

Validating classifier performance

- Classification performance is evaluated on a different set of samples with known class - the test set.
- The training set and the test set must be independent!
- Normally, the set of ground truth pixels (with known class) is partionioned into a set of training pixels and a set of test pixels of approximately the same size.
- This can be repeated several times to compute more robust estimates as average test accuracy over several different partitions of test set and training set.
  - By selecting e.g. 10 random partitions of the set of samples into a training set and a test set.

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

True class labels

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

Confusion matrix - cont.

Alternatives:

- Report nof. correctly classified pixels for each class.
- Report the percentage of correctly classified pixels for each class.
- Report the percentage of correctly classified pixels in total.
  - Why is this not a good measure if the number of test pixels from each class varies between classes?

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

A classification example

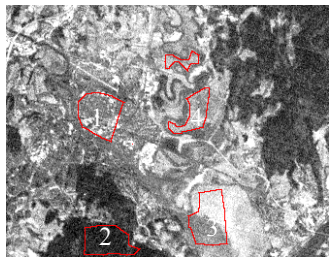
Landsat image with 6 spectral bands
The 6 bands will be the features
Training areas and test areas shown in mask

Upper part: RGB-false color image created from bands 4,5 and 6 with training and test regions overlaid.

Lower part: image of training regions only

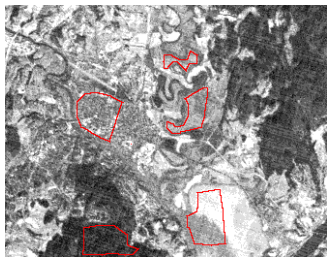
Visual inspection of feature 1

Class 2 (forest) seems to be well separated, Maybe also class 1 (urban)



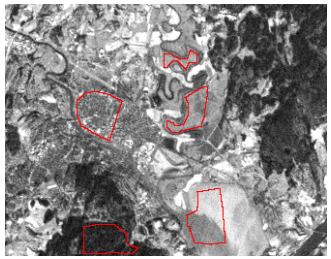
Visual inspection of feature 2

Class 2 (forest) seems to be well separated



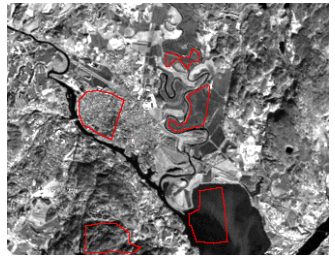
Visual inspection of feature 3

Class 2 (forest) seems to be well separated,
Class 1 (urban) seems to be well separated



Visual inspection of feature 4

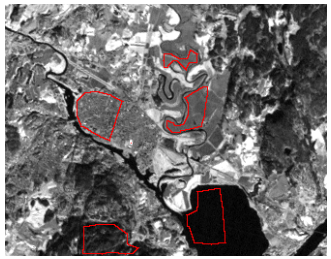
Class 1 (water) seems to be well separated,
Maybe also class 4 (agricultural)



Visual inspection of feature 5

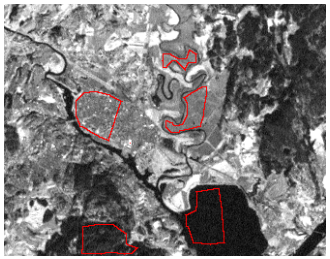
Water and forest appears similar - but the variance might be different

Urban and agricultural appears similar – but the variance might be different

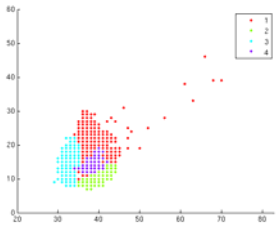


Visual inspection of feature 6

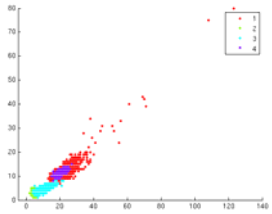
Seems similar to feature 5,
but with better contrast



Selected scatter plots (gscatter)

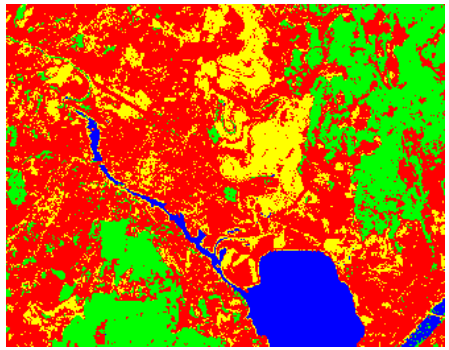


Scatterplot between feature 1 and 4



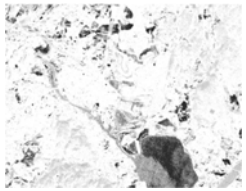
Scatterplot between feature 5 and 6

Classified images

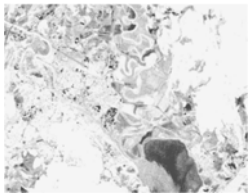


The entire image classified to the most probable class

Display the posterior probabilities as images



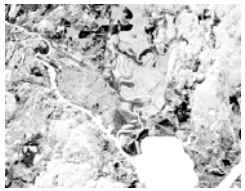
Posterior probability for class urban



Posterior probability for class forest

Dark values:
Probabilities close to 0

Bright values:
Probabilities close to 1



Posterior probability for class water

Posterior probability for class agricultural

Confusion matrix for the training set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1340	2	0	310
Class 1	43	1253	0	2
Class 3	0	0	1738	0
Class 4	131	3	0	1266

Accuracy per class: Averaged over all classes: 91.7%
Class1: 81%
Class2: 96%
Class3: 100%
Class4: 90%

Confusion matrix for the test set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1474	3	1	251
Class 1	513	2311	0	0
Class 3	14	0	1953	0
Class 4	213	2	0	1390

Accuracy per class: Averaged over all classes: 87.5%
Class1: 85%
Class2: 81%
Class3: 98%
Class4: 86%

Learning goals from this lecture

- Be able to **use and implement** Bayes rule with a d-dimensional Gaussian distribution.
- Know how μ_s and Σ_s are estimated.
- Understand the 2-dimensional case where a covariance matrix is illustrated as an ellipse.
- Be able to simplify the general discriminant function for 3 cases.
- Have a geometric interpretation of classification with 2 features.

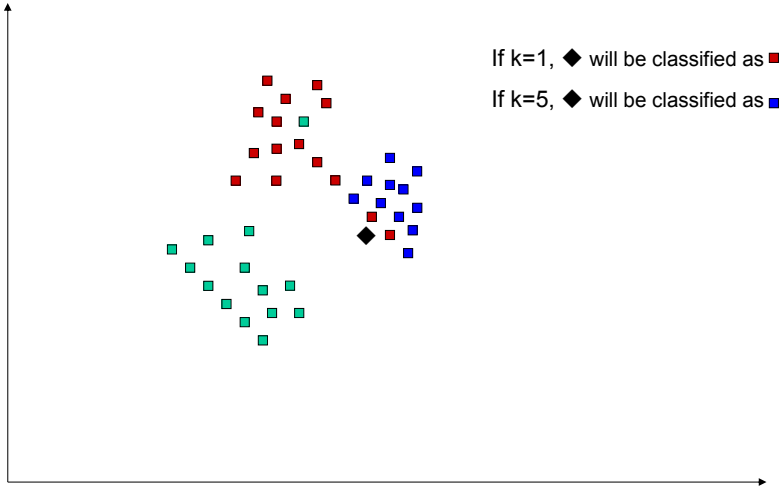
If time….

- The following slides are presented if time allows.
- Otherwise, they are presented next week.

k-Nearest-Neighbor classification

- A very simple classifier.
- Classification of a new sample x_j is done as follows:
  - Out of N training vectors, identify the k nearest neighbors (measured by Euclidean distance) in the training set, irrespectively of the class label.
  - 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)
  - Assign x_j to the class ω_i with the maximum number of k_i samples.
- k should be odd, and must be selected a priori.

kNN-example



About kNN-classification

- If *k=1* (1NN-classification), each sample is assigned to the same class as the closest sample in the training data set.
- If the number of training samples is very high, this can be a good rule.
- If k->∞, this is theoretically a very good classifier.
- This classifier involves no "training time", but the time needed to classify one pattern x_i will depend on the number of training samples, as the distance to all points in the training set must be computed.
- "Practical" values for *k*: *3<=k<=9*
- *Classification performance should **always** be computed on the test data set.*

Supervised or unsupervised classification

- Supervised classification
  - Classify each object or pixel into a set of *k* known classes
  - Class parameters are estimated using a set of training samples from each class.
- Unsupervised classification
  - Partition the feature space into a set of *k* clusters
  - *k* is not known and must be estimated (difficult)
- In both cases, classification is based on the value of the set of *n* features $x_1,....x_n$.
- The object is classified to the class which has the highest posterior probability.
- "The clusters we get are not the classes we want".

Unsupervised classification/clustering

- Divide the data into clusters based on similarity (or dissimilarity)
- Similarity or dissimilarity is based on distance measures (sometimes called proximity measures)
  - Euclidean distance, Mahalanobis distance etc.
- Two main approaches to clustering
  - hierarchical
    - divisive
    - agglomerative
  - non-hierarchical (sequential)
- Non-hierarachical methods are often used in image analysis

K-means clustering

- Note: K-means algorithm normally means ISODATA, but different definitions are found in different books
- K is assumed to be known

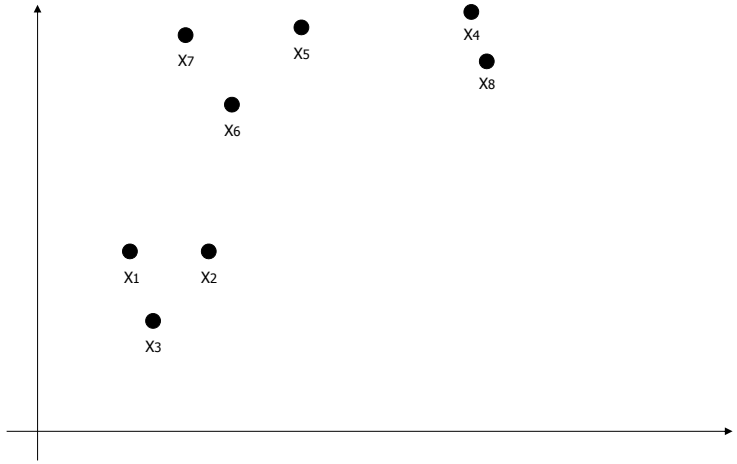
1. Start with assigning K cluster centers
   - k random data points, or the first K points, or K equally spaces points
   - For k=1:K, Set μ_k equal to the feature vector x_k for these points.
2. Assign each object/pixel x_i in the image to the closest cluster center using Euclidean distance.
   - Compute for each sample the distance r2 to each cluster center:

$$r^2 = (x_i - \mu_k)^T (x_i - \mu_k) = \|x_i - \mu_k\|^2$$

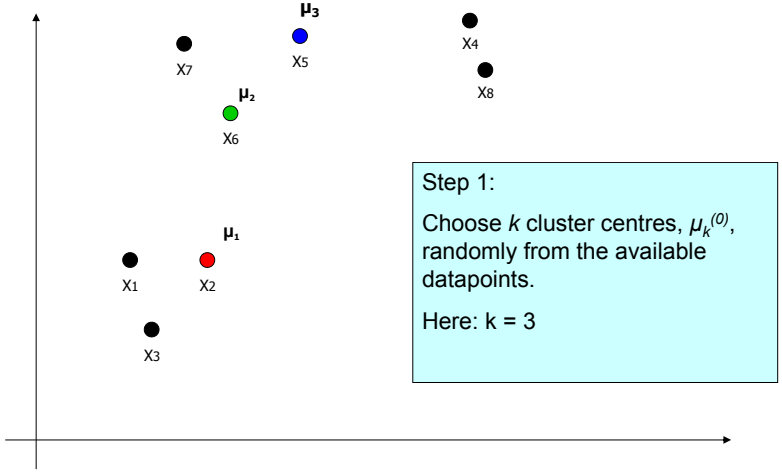
   - Assign x_i to the closest cluster (with minimum r value)
3. Recompute the cluster centers based on the new labels.
4. Repeat from 2 until #changes<limit.

ISODATA K-means: splitting and merging of clusters are included in the algorithm

k-means example



k-means example

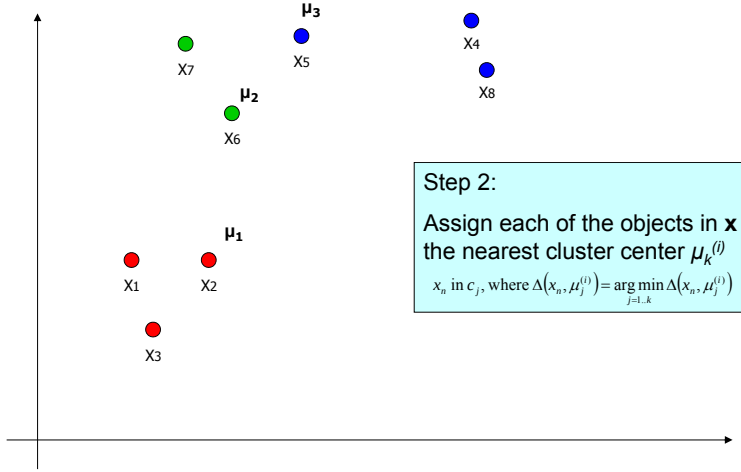


Step 1:

Choose k cluster centres, $\mu_k^{(0)}$, randomly from the available datapoints.

Here: k = 3

k-means example

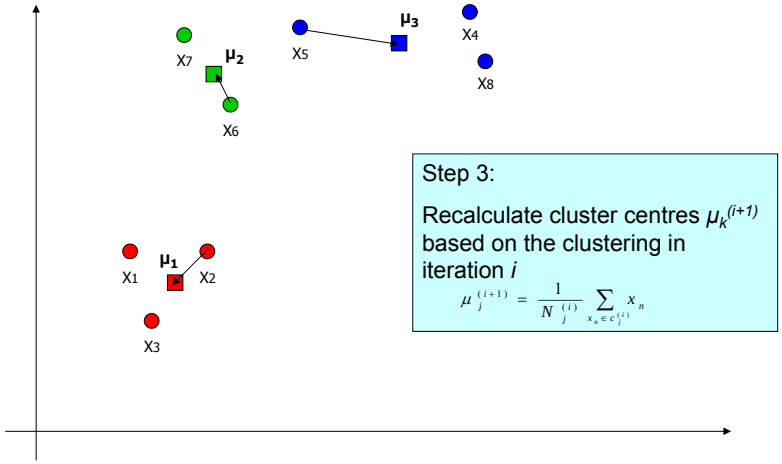


Step 2:

Assign each of the objects in **x** to the nearest cluster center $\mu_k^{(i)}$

$$x_n \text{ in } c_j, \text{ where } \Delta\left(x_n, \mu_j^{(i)}\right) = \arg\min_{j=1..k} \Delta\left(x_n, \mu_j^{(i)}\right)$$

k-means example

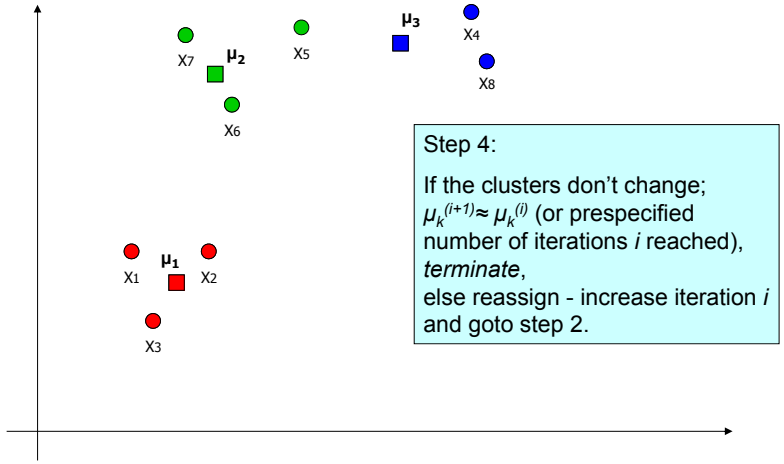


Step 3:

Recalculate cluster centres $\mu_k^{(i+1)}$ based on the clustering in iteration *i*

$$\mu_j^{(i+1)} = \frac{1}{N_j^{(i)}} \sum_{x_n \in c_j^{(i)}} x_n$$

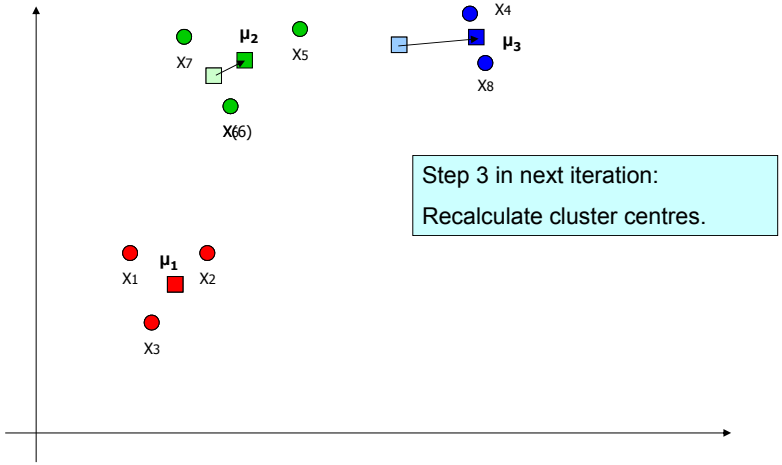
k-means example



Step 4:

If the clusters don't change; $\mu_k^{(i+1)} \approx \mu_k^{(i)}$ (or prespecified number of iterations *i* reached), *terminate*,
else reassign - increase iteration *i* and goto step 2.

k-means example



Step 3 in next iteration:

Recalculate cluster centres.

k-means variations

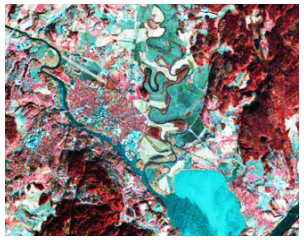
- The generic algorithm has many improvements
  - ISODATA – allow for merging and splitting of clusters
    - Among other things, this seeks to improve an initial "bad" choice of *k*
  - k-medians is another variation
  - k-means optimizes a probabilistic model

How do we determine k?

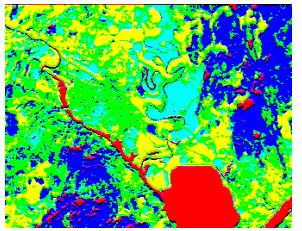
- The number of natural clusters in the data rarely corresponds to the number of information classes of interest.
- Cluster validity indices can give indications of how many clusters there are.
- Use cluster merging or splitting tailored to the application.
- Rule of thumb for practical image clustering:
  - start with approximately twice as many clusters as expected information classes
  - determine which clusters correspond to the information classes
  - split and merge clusters to improve.

Example: K-means clustering

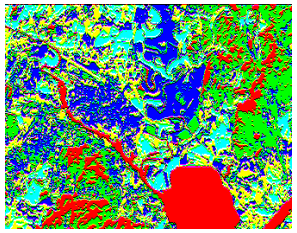
Original



Supervised
4 classes



Kmeans
K=5



Kmeans
K=10