INF 4300 – Digital Image Analysis

REPETITION

Anne Solberg 18.11.2015

Repetition -Erosion of a binary image Simplified notation

- To compute the erosion of pixel (x,y) in image f with the structuring element S: place the structuring elements such that its origo is at (x,y). Compute

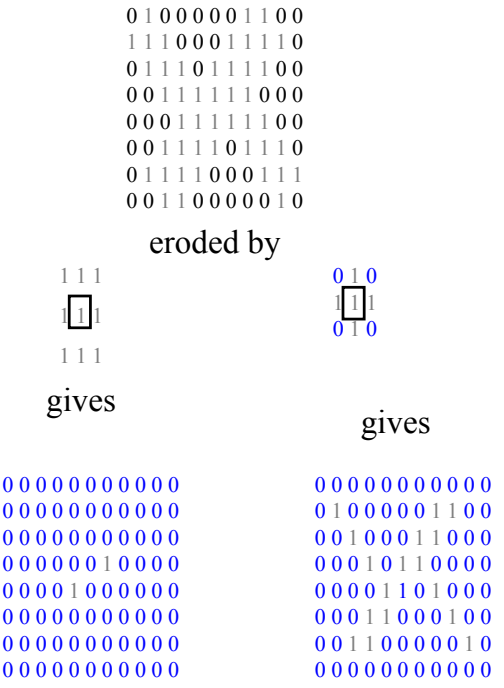
$$g(x,y)=\begin{cases}1 & \text{if S fits f}\\ 0 & \text{otherwise}\end{cases}$$

- Erosion of the image f with structuring element S is denoted

  ε (f|S) = f ⊖ S

- Erosion of a set A with the stucturing element B is defined as the position of all pixels x in A such that B is included in A when origo of B is at x.

$$A\theta B=\{x|B_x\subseteq A\}$$



Dilation of a binary image

- Place S such that origo lies in pixel (x,y) and use the rule

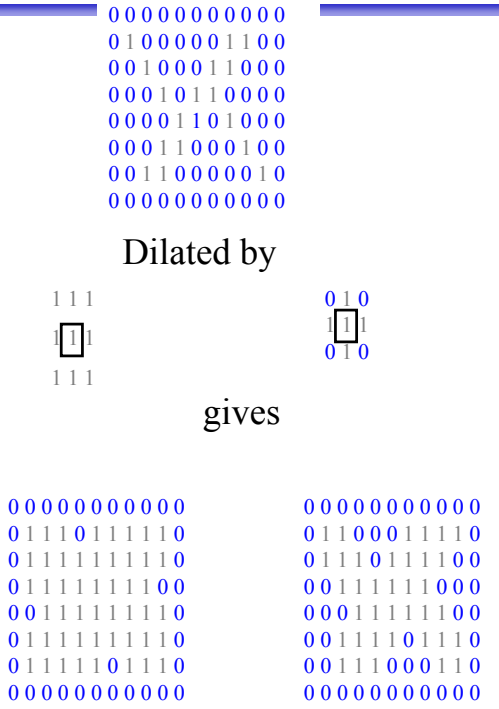
$$g(x,y) = \begin{cases} 1 & \text{if S hits f} \\ 0 & \text{otherwise} \end{cases}$$

- The image f dilated by the structuring element S is denoted:

$$f \oplus S$$

- Dilation of a set A with a structuring element B is defined as the position of all pixels x such that B overlaps with at least one pixel in A when the origin is placed at x.

$$A \oplus B = \{x \mid B_x \cap A \neq \emptyset\}$$



Opening

- Erosion of an image removes all structures that the structuring element can not fit inside, and shrinks all other structures.
- Dilating the result of the erosion with the same structuring element, the structures that survived the erosion (were shrunken, not deleted) will be restored.
- This is called morphological opening:

$$f \circ S = (f \ominus S) \oplus S$$

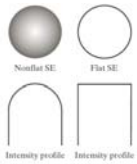
- The name tells that the operation can create an opening between two structures that are connected only in a thin bridge, without shrinking the structures (as erosion would do).

Closing

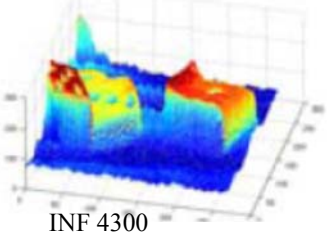
- A dilation of an object grows the object and can fill gaps.
- If we erode the result with the rotated structuring element, the objects will keep their structure and form, but small holes filled by dilation will not appear.
- Objects merged by the dilation will not be separated again.
- Closing is defined as $f \bullet S = \left(f \oplus \hat{S}\right) \theta \, \hat{S}$
- This operation can close gaps between two structures without growing the size of the structures like dilation would.

Gray level morphology

- We apply a simplified definition of morphological operations on gray level images
  - Grey-level erosion, dilation, opening, closing
- Image f(x,y)
- Structuring element b(x,y)
  - Nonflat or flat

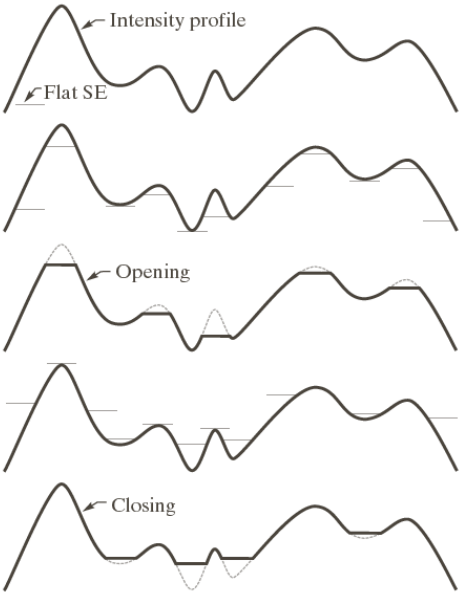


- Assume symmetric, flat structuring element, origo at center (this is sufficient for normal use).
- Erosion and dilation then correspond to local minimum and maximum over the area defined by the structuring element



Interpretation of grey-level opening and closing

- Intensity values are interpreted as height curves over the (x,y)-plane.
- Opening of f by b:
  Push the structuring element up from below towards the curve f.
  The value assigned is the highest level b can reach.
  smooths bright values down.
- Closing:
  Push the structuring element down from above towards the curve f.
  smooths dark values upwards



Top-hat transformation

- Purpose: detect (or remove) structures of a certain size.
- Top-hat: detects light objects on a dark background
  - also called white top-hat.
- Bottom-hat: detects dark objects on a bright background
  - also called black top-hat.
- Top-hat:

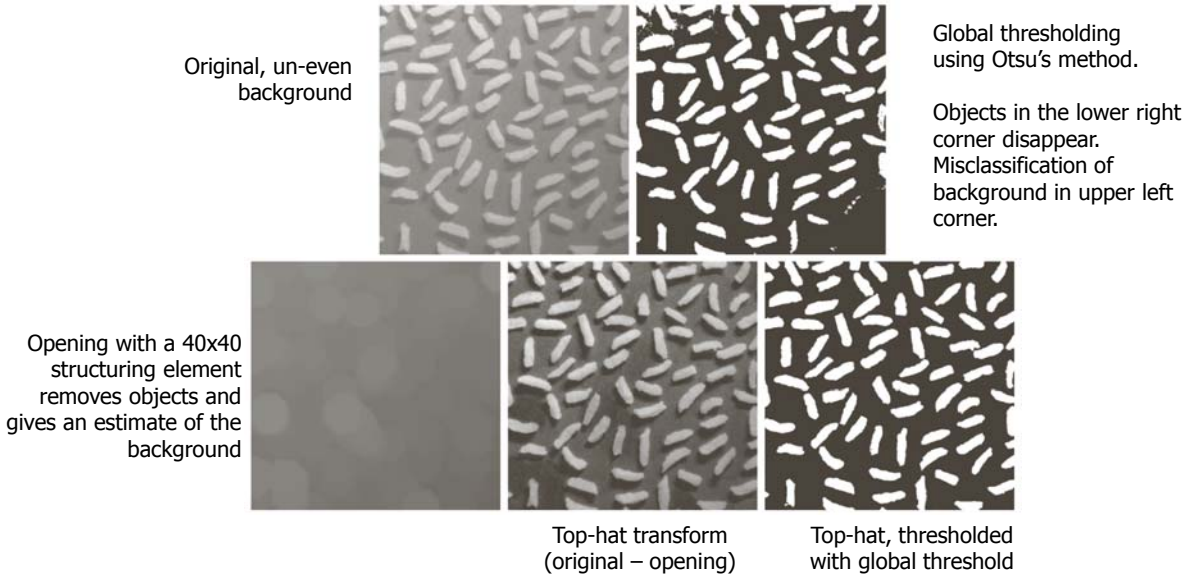
$$f - (f \circ b)$$

- Bottom-hat:

$$(f \bullet b) - f$$

- Very useful for correcting uneven illumination/objects on a varying background ☺

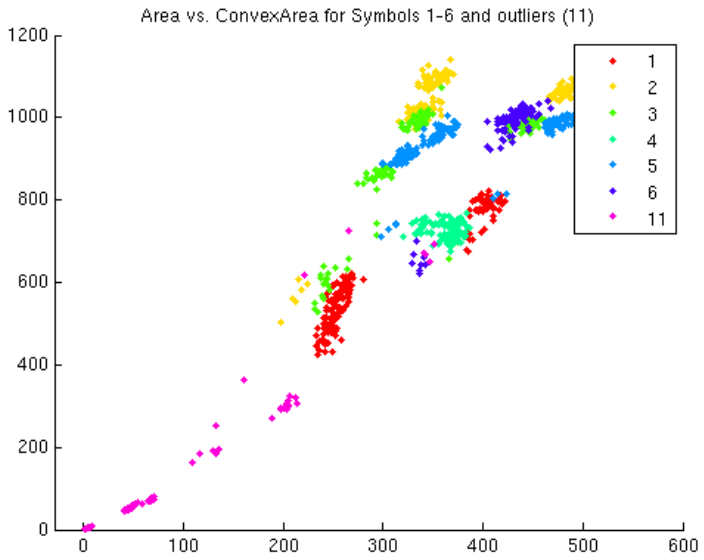
Example – top-hat



Learning goals - morphology

- Understand in detail binary morphological operations and selected applications:
  - Basic operators (erosion, dilation, opening, closing)
    - Understand the mathematical definition, perform them "by hand" on new objects
  - Applications of morphology:
    - edge detection, connected components, convex hull etc.
    - Verify the examples in the book
- Grey-level morphology:
  - Understand how grey-level erosion and dilation (and opening and closing) works.
  - Understand the effect these operations have on images.
  - Understand top-hat, bottom-hat and what they are used for.

Two correlated features – features align on a line



Mean vectors and covariance matrices in N dimensions

- If f(**x**) is a n-dimensional feature vector, we can formulate its mean vector and covariance matrix as:

$$f(\mathbf{x}) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ . \\ . \\ f_n(x) \end{bmatrix} \qquad \boldsymbol{\mu} = E[\mathbf{x}] = \begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_n) \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ \mu_n \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{n1} & \sigma_{n2} & . & . & \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & . & . & \sigma_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{n1} & \sigma_{n2} & . & . & \sigma_n^2 \end{bmatrix}$$

with n features, the mean vector μ will be of size 1xn and Σ or size nxn.

- The matrix will be symmetric as $\sigma_{kl} = \sigma_{lk}$

Bayes rule for a classification problem

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

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

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

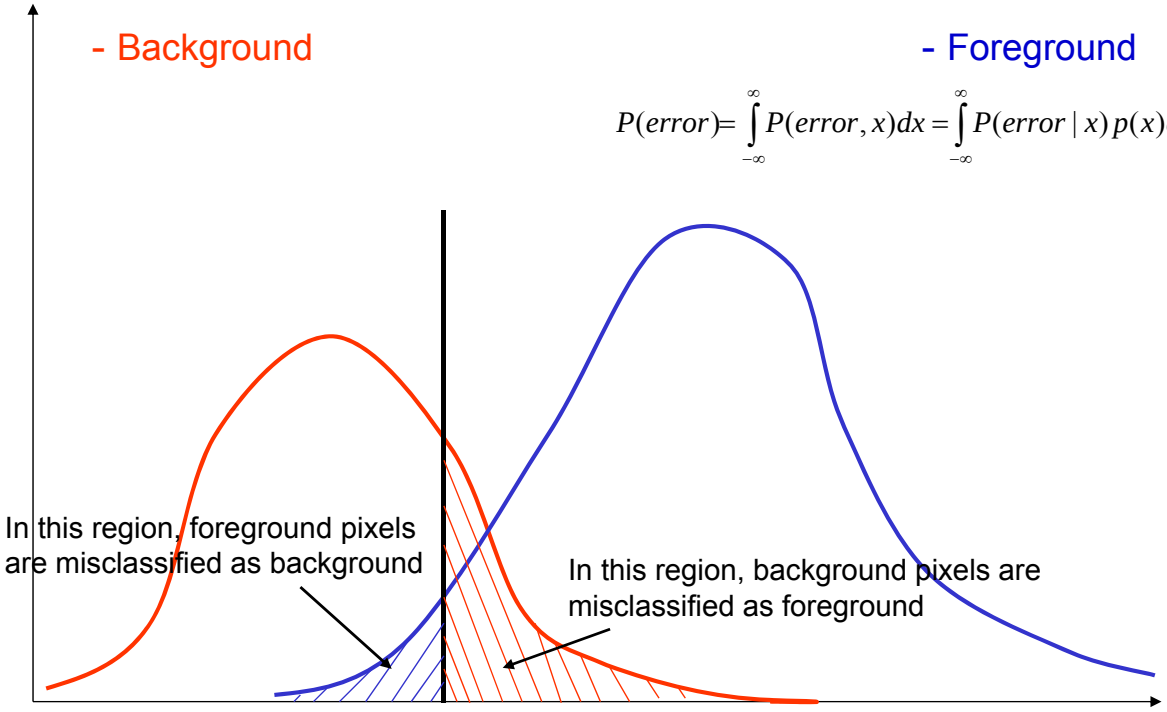
- For thresholding, $P(\omega_j)$ is the prior probability for background or foreground. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).
- Small p means a probability distribution
- Capital P means a probability (scalar value between 0 and 1)

Probability of error

- If we have 2 classes, we make an error either if we decide ω_1 if the true class is ω_2 if we decide ω_2 if the true class is ω_1.
- If $P(\omega_1|x) > P(\omega_2|x)$ we have more belief that x belongs to ω_1, and we decide ω_1.
- The probability of error is then:

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) \text{ if we decide } \omega_2 \\ P(\omega_2 \mid x) \text{ if we decide } \omega_1 \end{cases}$$

Back to classification error for thresholding



Minimizing the error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$

- When we derived the optimal threshold, we showed that the minimum error was achieved for placing the threshold (or *decision boundary* as we will call it now) at the point where

  $P(\omega_1|x) = P(\omega_2|x)$

- This is still valid.

Bayes decision rule

- In the 2 class case, our goal of minimizing the error implies a decision rule:

  Decide ω_1 if $P(\omega_1|x)>P(\omega_2|x)$; otherwise ω_2
- For J classes, the rule analogusly extends to choose the class with *maximum a posteriori* probability
- The *decision boundary* is the"border" between classes i and j, simply where $P(\omega_i|x)=P(\omega_j|x)$
  - Exactly where the threshold was set in minimum error thresholding!

Discriminant functions

- The decision rule

  Decide ω_1 if $P(\omega_1 \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x})$, for all $j \neq i$

  can be written as assign $\mathbf{x}$ to ω_1 if

  $$g_i(\mathbf{x}) > g_j(\mathbf{x})$$
- The classifier computes J discriminant functions $g_i(\mathbf{x})$ and selects the class corresponding to the largest value of the discriminant function.
- Since classification consists of choosing the class that has the largest value, a scaling of the discriminant function $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$ will not effect the decision if f is a monotonically increasing function.
- This can lead to simplifications as we will soon see.

Equivalent discriminant functions

- The following choices of discriminant functions give equivalent decisions:

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

- The effect of the decision rules is to divide the feature space into c decision regions R_1,.....R_c.
- If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is in region R_i.
- The regions are separated by decision boundaries, surfaces in features space where the discriminant functions for two classes are equal

The conditional density $p(\mathbf{x}|\ \omega_s)$

- Any probability density function can be used to model $p(\mathbf{x}|\ \omega_s)$
- A common model is the multivariate Gaussian density.
- The multivariate Gaussian density:

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

- 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

Inspecting p(**x**|ωs)

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

Scalar probability

Scalar

1xn vector transposed

nx1 vector transposed

nxn matrix
Inverse of covariance matrix

The mean vectors μs 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

The covariance matrix Σ_s for each class

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

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

- with d features, the covariance matrix Σ_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 Σ_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

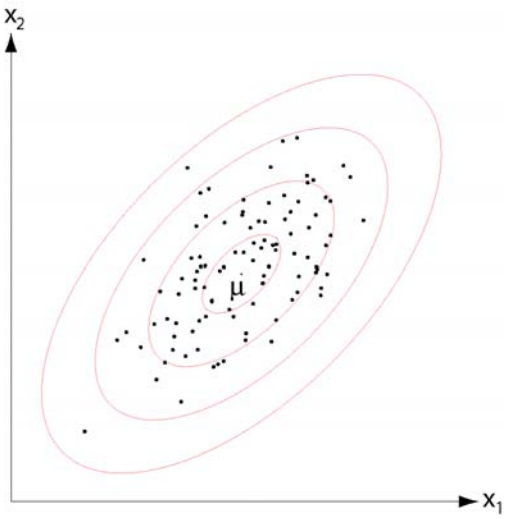
- The covariance matrix Σ_s will always be symmetric and positive semidefinite.
- If all components of x have non-zero variance, Σ_s will be positive definite.
- σ_{ij} is the covariance between features *i* and *j*.
- If features x_i and x_j are uncorrelated, $\sigma_{ij} = 0$.
- In the general case,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



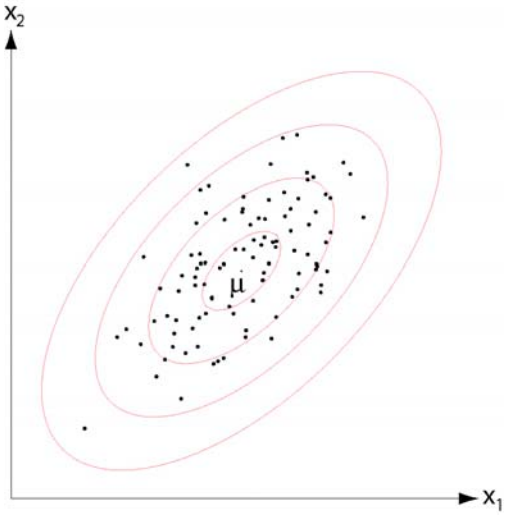
The covariance matrix and ellipses

- 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:

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

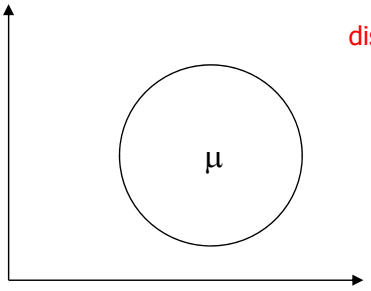


The main axes of the ellipse is determined by the eigenvectors of Σ. The eigenvalues of Σ gives their length.

Euclidean distance vs. Mahalanobis distance

- Euclidean distance between point x and class center μ:

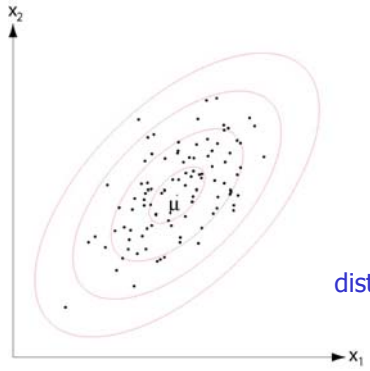
$$(x-\mu)^T(x-\mu)=\|x-\mu\|^2$$



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

- We saw last lecture that the minimum-error-rate classification can be computed using the discriminant functions

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

- 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: $\Sigma_j=\sigma^2 I$

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

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

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

Common for all classes, no need to compute these terms

Since $\boldsymbol{x^T x}$ is common for all classes, an equivalent $g_j(\boldsymbol{x})$ is a linear function of $\boldsymbol{x}$:.

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

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

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

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_i 0$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } wi0 = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

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

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

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

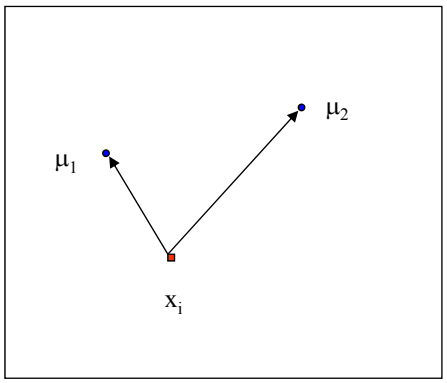
$$\text{and } x_0 = \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) - \frac{\sigma^2}{\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

- $\mathbf{w}=\mu_i-\mu_j$ 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

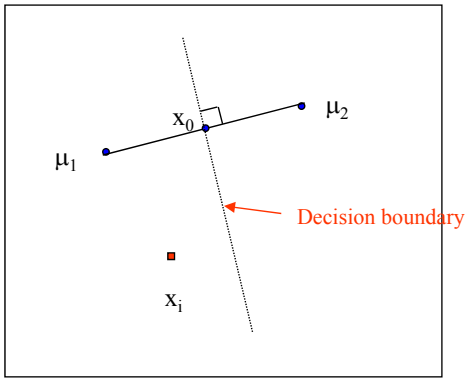
- If the features were indepenent ($\Sigma_j=\sigma^2 I$) the discriminant function was simplified to:

$$g_j'(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) + \ln P(\omega_j)$$
$$= -\frac{1}{2\sigma^2}\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|^2 + \ln P(\omega_j)$$

- 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_s 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.



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 $\boldsymbol{x^Tx}$ 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.
- The decision boundary has the equation:

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

$$\mathbf{w} = \Sigma^{-1}(\mu_i - \mu_j)$$

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln\left[(P(\omega_i)/(P(\omega_j)\right]}{(\mu_i-\mu_j)^T\Sigma^{-1}(\mu_i-\mu_j)}(\mu_i - \mu_j)$$

- Because $\mathbf{w}_i = \boldsymbol{\Sigma}^{-1}(\mu_i - \mu_j)$ is not in the direction of $(\mu_i - \mu_j)$, the hyperplane will not be orthogonal to the line between the means.

Case 3:, Σ_j=arbitrary

- The discriminant functions will be quadratic:

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

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

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

- 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.

Confusion matrices

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

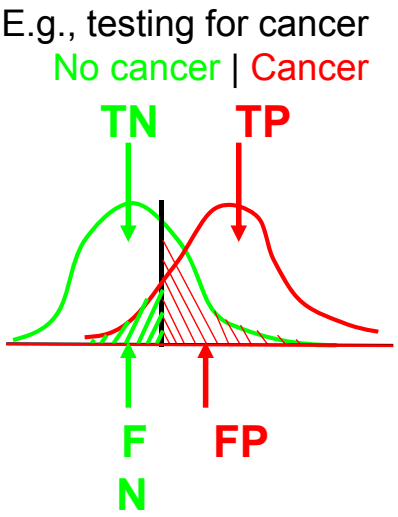
Estimated class labels

True class labels

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

True / False positives / negatives

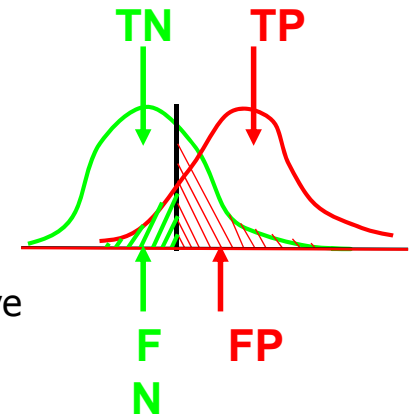
- **True positive (TP):**
  Patient has cancer
  and test result is positive.
- True negative (TN):
  A healthy patient
  and a negative test result.
- **False positive (FP):**
  Healthy patient that gets a positive test result.
- **False negative (FN):**
  Cancer patient that gets a negative test result.



- *Good to have: **TP** & **TN***
- *Bad to have: **FP** (but this will probably be detected)*
- *Worst to have: **FN** **(may go un-detected)***

Sensitivity and specificity

- ***Sensitivity:***
  the portion of the data set that tested positive out of all the positive patients tested:
  - **Sensitivity = TP/(TP+FN)**
  - The probability that the test is positive given that the patient is sick.
  - Higher sensitivity means that fewer decease cases go undetected.
- ***Specificity:***
  the portion of the data set that tested negative out of all the negative patients tested:
  - **Specificity = TN/(TN+FP)**
  - The probability that a test is negative given that the patient is not sick.
  - Higher specificity means that fewer healthy patients are labeled as sick.



Outliers and doubt

- In a classification problem, we might want to identify outliers and doubt samples
- We might want an ideal classifier to report
  - 'this sample is from class l' (usual case)
  - 'this sample is not from any of the classes' (outlier)
  - 'this sample is too hard for me' (doubt/reject)
- The two last cases should lead to a rejection of the sample!

The covariance matrix and dimensionality

- Assume we have S classes and a d-dimensional feature vector.
- With a fully multivariate Gaussian model, we must estimate S different mean vectors and S different covariance matrices from training samples.

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

$\hat{\Sigma}_s$ has d(d+1)/2 elements

- Assume that we have M_s training samples from each class
- Given M_s, there is a maximum of the achieved classification performance for a certain value of d
  - increasing n beyond this limit will lead to worse performance.
- Adding more features is not always a good idea!
- Total number of samples given by a rule of thumb: **M>10 d S**
- If we have limited training data, we can use diagonal covariance matrices or regularization

The ”curse” of dimensionality

- In practice, the curse means that, for a given sample size, there is a maximum number of features one can add before the classifier starts to degrade.

- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- For a high dimensionality, we will need lots of training data to get the best performance.
- => ≈10 samples / feature / class.

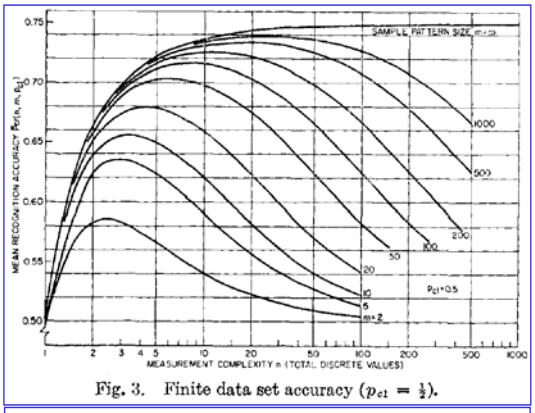


Fig. 3. Finite data set accuracy ($p_{ct} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

Use few, but good features

- To avoid the ”curse of dimensionality” we must take care in finding a set of relatively few features.
- A good feature has high within-class homogeneity, and should ideally have large between-class separation.
- In practise, one feature is not enough to separate all classes, but a good feature should:
  - separate some of the classes well
  - Isolate one class from the others.
- If two features look very similar (or have high correlation), they are often redundant and we should use only one of them.
- Class separation can be studied by:
  - Visual inspection of the feature image overlaid the training mask
  - Scatter plots
- Evaluating features as done by training can be difficult to do automatically, so manual interaction is normally required.

How do we beat the ”curse of dimensionality”?

- Use regularized estimates for the Gaussian case
  - Use diagonal covariance matrices
  - Apply regularized covariance estimation
- Generate few, but informative features
  - Careful feature design given the application
- Reducing the dimensionality
  - Feature selection – select a subset of the original features (more in INF5300)
  - Feature transforms – compute a new subset of features based on a linear combination of all features (INF 5300)
    - Example 1: Principal component transform
      - Unsupervised, finds the combination that maximized the variance in the data.
    - Example 2: Fisher's linear discriminant
      - Supervised, finds the combination that maximizes the distance between the classes.

Distance measures used in feature selection

- In feature selection, each feature combination must be ranked based on a criterion function.
- Criteria functions can either be distances between classes, or the classification accuracy on a validation test set.
- If the criterion is based on e.g. the mean values/covariance matrices for the training data, distance computation is fast.
- Better performance at the cost of higher computation time is found when the classification accuracy on a validation data set (different from training and testing) is used as criterion for ranking features.
  - This will be slower as classification of the validattion data needs to be done for every combination of features.

Distance measures between classes

- How do be compute the distance between two classes:
  - Distance between the closest two points?
  - Maximum distance between two points?
  - Distance between the class means?
  - Average distance between points in the two classes?
  - Which distance measure?
    - Euclidean distance or Mahalanobis distance?
- Distance between K classes:
  - How do we generalize to more than two classes?
  - Average distance between the classes?
  - Smallest distance between a pair of classes?

Class separability measures

- How do we get an indication of the separability between two classes?
  - Euclidean distance between class means $|\mu_r - \mu_s|$
  - Bhattacharyya distance
    - Can be defined for different distributions
    - For Gaussian data, it is

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln \frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

  - Mahalanobis distance between two classes:

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$
$$\Sigma = N_1\Sigma_1 + N_2\Sigma_2$$

Method 2 - Sequential backward selection

- Select l features out of d
- Example: 4 features x_1,x_2,x_3,x_4
- Choose a criterion C and compute it for the vector $[x_1,x_2,x_3,x_4]^T$
- Eliminate one feature at a time by computing $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$
- Select the best combination, say $[x_1,x_2,x_3]^T$.
- From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched:
  $1+1/2((d+1)d-l(l+1))$

Method 3: Sequential forward selection

- Compute the criterion value for each feature. Select the feature with the best value, say x_1.
- Form all possible combinations of features x1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]^T$, $[x_1,x_3]^T$, $[x_1,x_4]^T$, etc. Compute the criterion and select the best one, say $[x_1,x_3]^T$.
- Continue with adding a new feature.
- Number of combinations searched: ld-l(l-1)/2.
  - Backwards selection is faster if l is closer to d than to 1.

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.

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_j 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_j 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