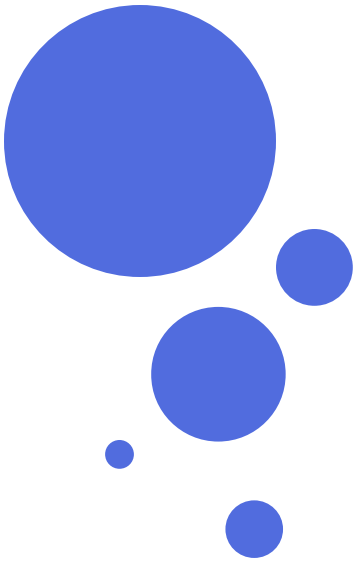
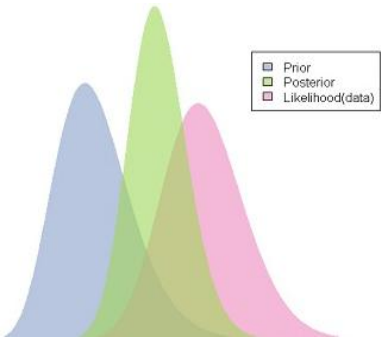
Rensselaer

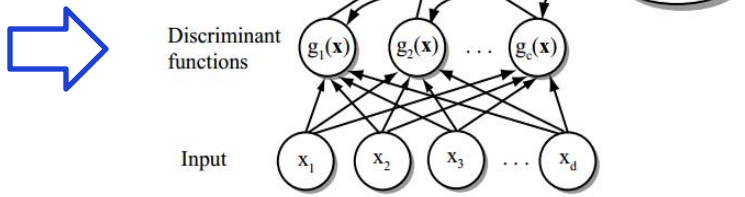


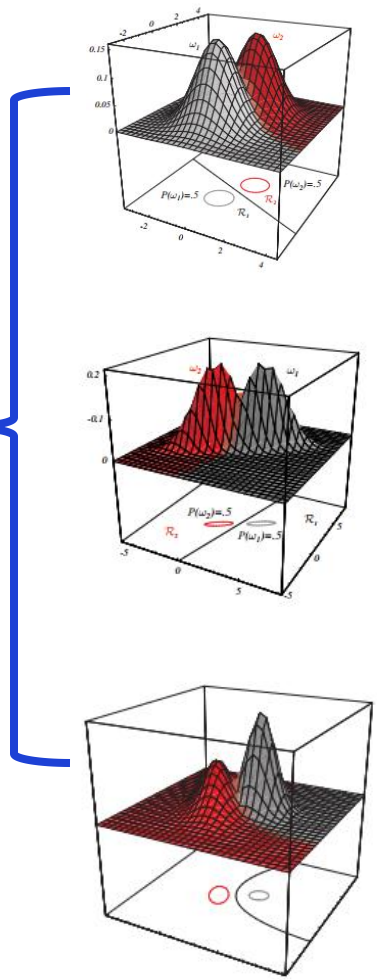
Lecture 4: Bayesian Decision Theory and Max Likelihood Estimation

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

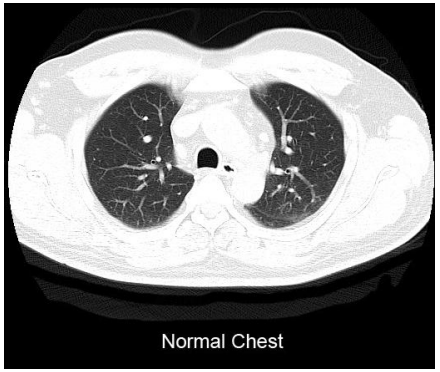
Recap Previous Lecture

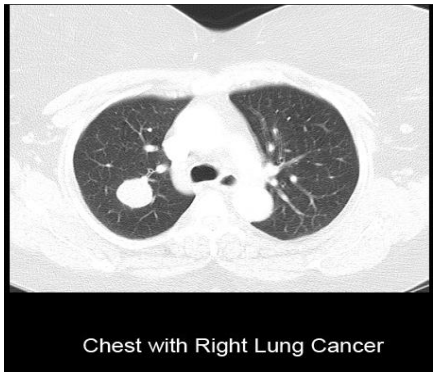






Recap Previous Lecture





From a medical image, we want to classify (determine) whether it contains cancer tissues or not.

$$\lambda(\alpha_i|\omega_j)$$

	cancer ω_1	normal ω_2
cancer α_1	0	1
normal α_2	100	0

$$\omega_1 = \text{cancer}, \quad \omega_2 = \text{normal}$$
$$\alpha_1 = \text{cancer}, \quad \alpha_2 = \text{normal}$$

Ground truths is always unknown for classifiers.

$$R(a_i/\mathbf{x}) = \sum_{j=1}^{c} \lambda(a_i/\omega_j) P(\omega_j/\mathbf{x})$$

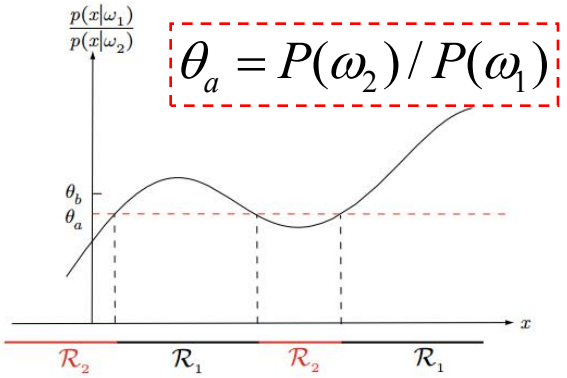
$$R(a_1/\mathbf{x}) = \lambda_{11}P(\omega_1/\mathbf{x}) + \lambda_{12}P(\omega_2/\mathbf{x})$$
$$R(a_2/\mathbf{x}) = \lambda_{21}P(\omega_1/\mathbf{x}) + \lambda_{22}P(\omega_2/\mathbf{x})$$

$$R(a_1/\mathbf{x}) < R(a_2/\mathbf{x})$$

$$\frac{p(\mathbf{x}/\omega_1)}{p(\mathbf{x}/\omega_2)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})}\frac{P(\omega_2)}{P(\omega_1)}$$

$$\theta_b = \frac{P(\omega_2)(\lambda_{12}-\lambda_{22})}{P(\omega_1)(\lambda_{21}-\lambda_{11})}$$

$$\theta_a = P(\omega_2)/P(\omega_1)$$



Outline

- Bayesian Decision Theory
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- Example with Real World Data

Outline

- **Bayesian Decision Theory**
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- Example with Real World Data

Error Bounds

- Exact error calculations could be difficult – easier to estimate error bounds!

$$P(error) = \int P(error, \mathbf{x})\, d\mathbf{x} = \int P(error/\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}$$

or

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

$\min[P(\omega_1/x), P(\omega_2/x)]$

Using the inequality:

$$min[a, b] \le a^{\beta} b^{1-\beta}, \qquad a, b \ge 0, 0 \le \beta \le 1$$

$$P(\text{error}) = \int min[p(\mathbf{x}/\omega_1)P(\omega_1),\ p(\mathbf{x}/\omega_2)P(\omega_2)]d\mathbf{x} \le$$

$$P^{\beta}(\omega_1)P^{1-\beta}(\omega_2) \int p^{\beta}(\mathbf{x}/\omega_1)\, p^{1-\beta}(\mathbf{x}/\omega_2)d\mathbf{x} = e^{-\kappa(\beta)}$$

Error Bounds

$$P(\text{error}) = \int min[p(\mathbf{x}/\omega_1)P(\omega_1),\ p(\mathbf{x}/\omega_2)P(\omega_2)]d\mathbf{x} \leq$$

$$P^{\beta}(\omega_1)P^{1-\beta}(\omega_2)\int p^{\beta}(\mathbf{x}/\omega_1)\ p^{1-\beta}(\mathbf{x}/\omega_2)d\mathbf{x} = e^{-\kappa(\beta)}$$

- If the class conditional distributions are Gaussian, then

$$\int p^{\beta}(\mathbf{x}|\omega_1)p^{1-\beta}(\mathbf{x}|\omega_2)\ d\mathbf{x} = e^{-k(\beta)}$$

where

$$k(\beta) = \frac{\beta(1-\beta)}{2}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^t[\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2]^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)+\frac{1}{2}\ln\frac{|\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|^{\beta}|\boldsymbol{\Sigma}_2|^{1-\beta}}.$$

Error Bounds

- The *Chernoff* bound is obtained by minimizing $e^{\kappa(\beta)}$
  - This is a 1-D optimization problem, regardless to the dimensionality of the class conditional densities.

$$k(\beta) = \frac{\beta(1-\beta)}{2}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^t[\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2]^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)+\frac{1}{2}\ln\frac{|\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|^{\beta}|\boldsymbol{\Sigma}_2|^{1-\beta}}.$$

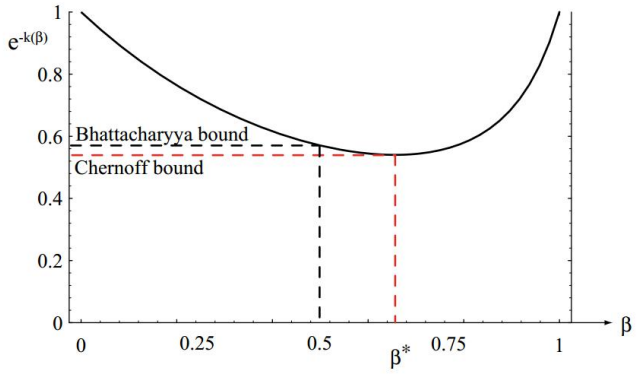


Figure 2.18: The Chernoff error bound is never looser than the Bhattacharyya bound. For this example, the Chernoff bound happens to be at $\beta^* = 0.66$, and is slightly tighter than the Bhattacharyya bound ($\beta = 0.5$).

Error Bounds

- The *Bhattacharyya* bound is obtained by setting $\beta = 0.5$ Easier to compute than Chernoff error but looser.

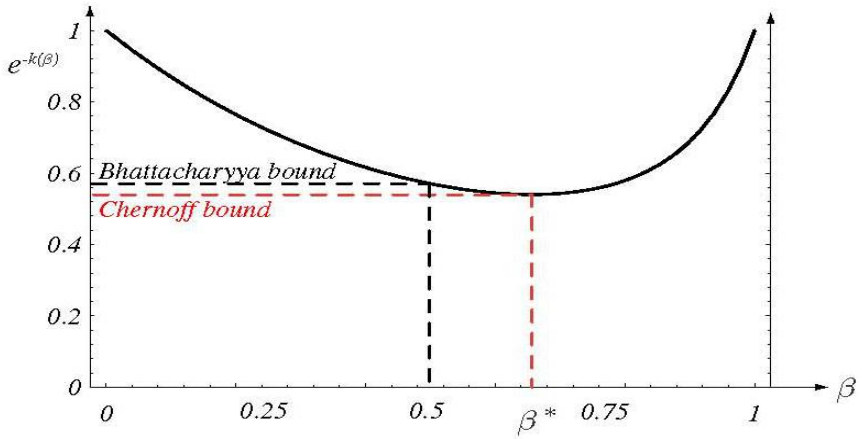


FIGURE 2.18. The Chernoff error bound is never looser than the Bhattacharyya bound. For this example, the Chernoff bound happens to be at $\beta^* = 0.66$, and is slightly tighter than the Bhattacharyya bound ($\beta = 0.5$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

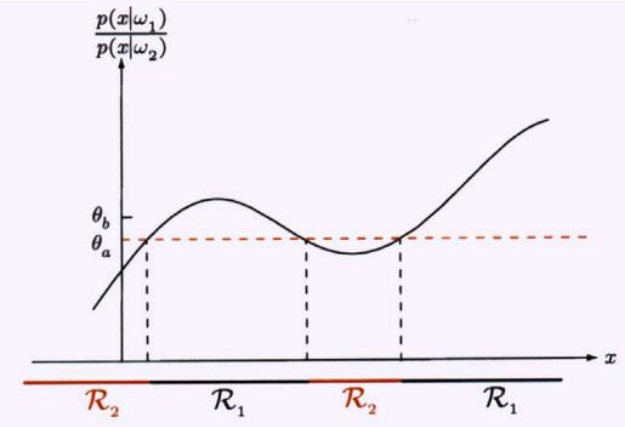
- **Note**: the Chernoff and Bhattacharyya bounds will not be good bounds if the densities are not Gaussian.

Outline

- **Bayesian Decision Theory**
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- Example with Real World Data

Receiver Operating Characteristic (ROC) Curve

- Every classifier typically employs some kind of a threshold.



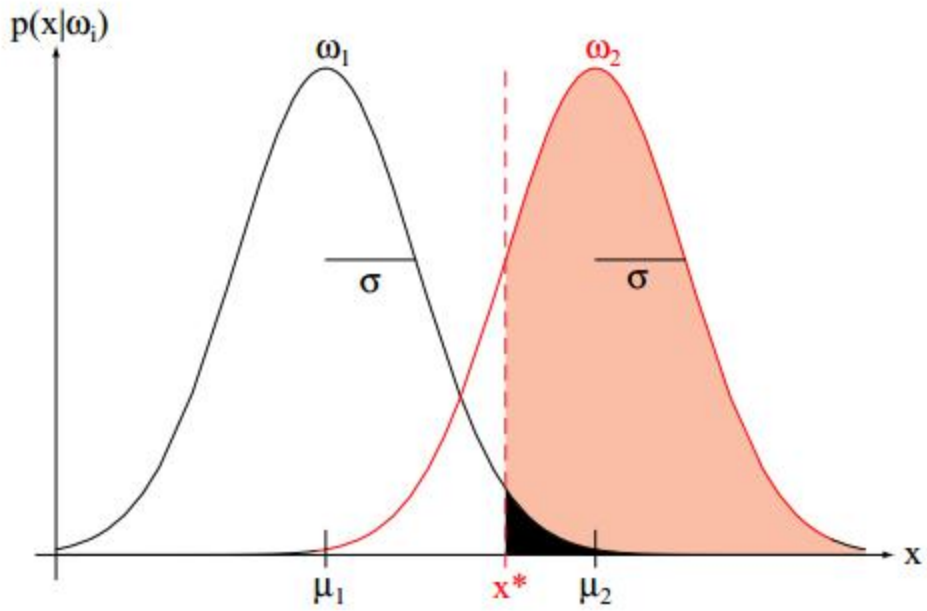
$$\theta_a = P(\omega_2)/P(\omega_1)$$

$$\theta_b = \frac{P(\omega_2)(\lambda_{12}-\lambda_{22})}{P(\omega_1)(\lambda_{21}-\lambda_{11})}$$

- Changing the threshold will affect the performance of the classifier.
- ROC curves allow us to evaluate the performance of a classifier using different thresholds.

Example: Person Authentication

- Authenticate a person using biometrics (e.g., fingerprints).
- There are two possible distributions (i.e., classes):



Example: Person Authentication

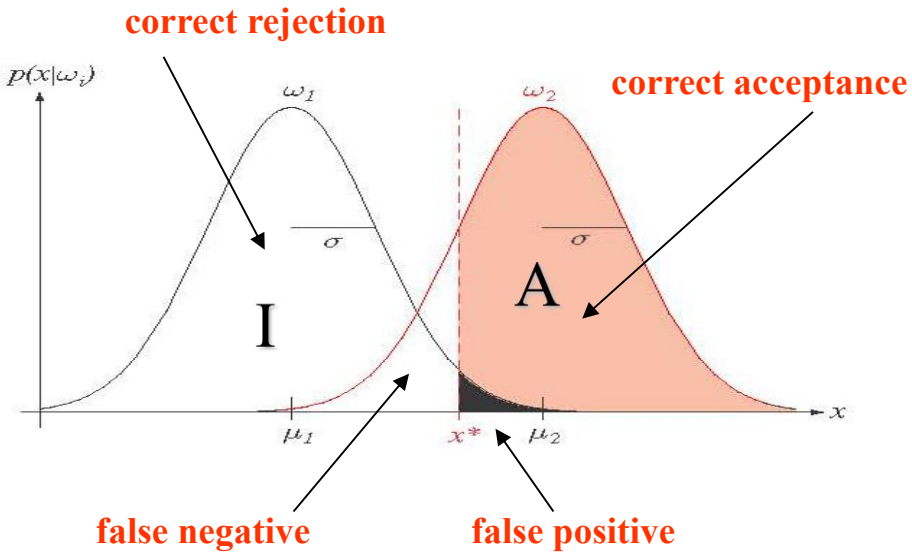
- Possible decisions:

  (1) **correct acceptance** (true positive): X belongs to A, and we decide A

  (2) **incorrect acceptance** (false positive): X belongs to I, and we decide A

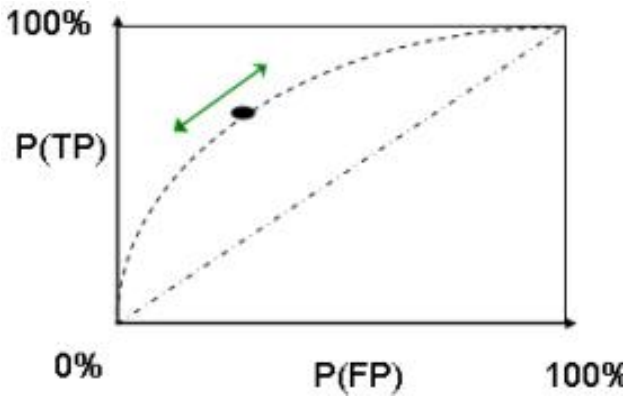
  (3) **correct rejection** (true negative): X belongs to I, and we decide I

  (4) **incorrect rejection** (false negative): X belongs to A, and we decide I

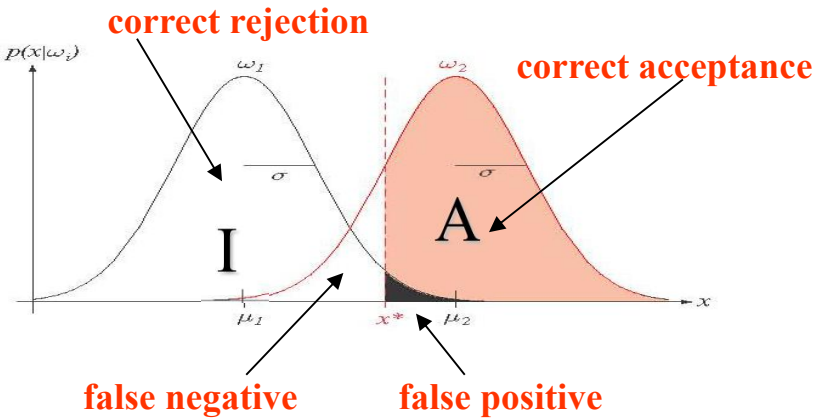


$$TPR = TP/P = TP/(TP + FN)$$

$$FPR = FP/N = FP/(FP + TN)$$

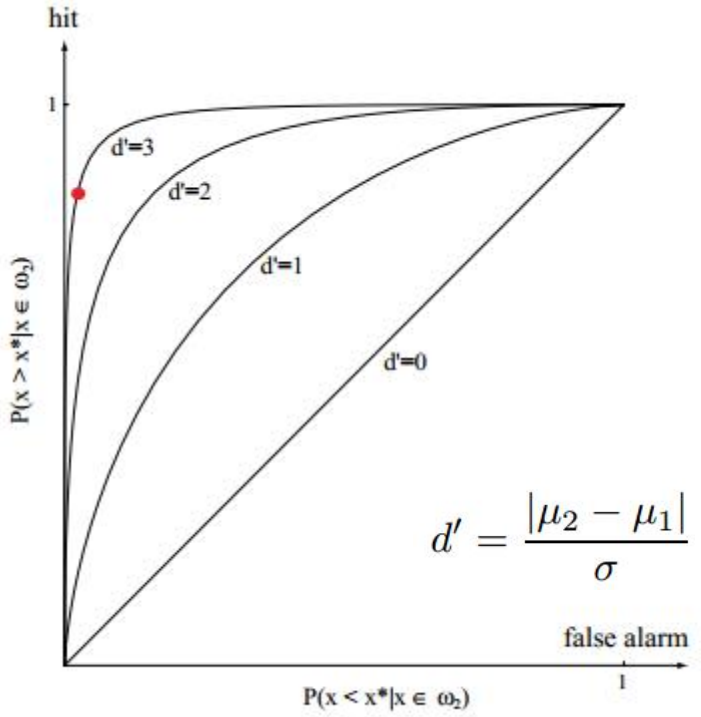


ROC Curve



$$TPR = TP/P = TP/(TP + FN)$$

$$FPR = FP/N = FP/(FP + TN)$$



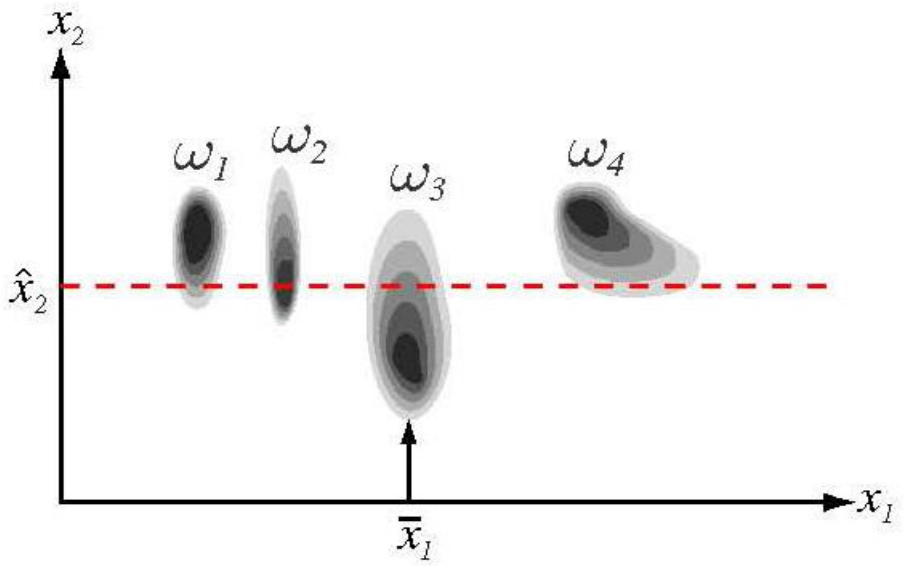
FPR: False Positive Rate (X-axis)
TRR: True Postive Rate (Y-axis)

Outline

- **Bayesian Decision Theory**
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- Example with Real World Data

Missing Features

- Suppose **x**=(x_1,x_2) is a test vector where x_1 is missing and via $x_2 = \hat{x}_2$ how can we classify it?
  - If we set x_1 equal to the average value, we will classify **x** as ω_3
  - But $p(\hat{x}_2 / \omega_2)$ is larger; should classify **x** as ω_2 ?



Missing Features

- Suppose $\mathbf{x}=[\mathbf{x}_g, \mathbf{x}_b]$ ($\mathbf{x}_g$: good features, $\mathbf{x}_b$: bad features)
- Derive the Bayes rule using the good features:

$$P(\omega_i/\mathbf{x}_g) = \frac{P(\omega_i, \mathbf{x}_g)}{p(\mathbf{x}_g)} = \frac{\int P(\omega_i, \mathbf{x}_g, \mathbf{x}_b)d\mathbf{x}_b}{p(\mathbf{x}_g)} =$$

$$\frac{\int P(\omega_i/\mathbf{x}_g, \mathbf{x}_b)p(\mathbf{x}_g, \mathbf{x}_b)d\mathbf{x}_b}{p(\mathbf{x}_g)} = \frac{\int P(\omega_i/\mathbf{x}_g, \mathbf{x}_b)p(\mathbf{x})d\mathbf{x}_b}{\int p(\mathbf{x})d\mathbf{x}_b}$$

marginalize posterior probability over bad features.

Decide ω_1 if $P(\omega_1/\mathbf{x}_g) > P(\omega_2/\mathbf{x}_g)$; otherwise decide ω_2

Outline

- **Bayesian Decision Theory**
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- Example with Real World Data

Compound Bayesian Decision Theory

- Sequential decision
  - Decide as each pattern (e.g., fish) emerges.

- Compound decision
  - Wait for *n* patterns (e.g., fish) to emerge.
  - Make all *n* decisions jointly.

  Could improve performance when consecutive states of nature are **not** be statistically independent.

Compound Bayesian Decision Theory

- Suppose $\boldsymbol{\omega} = (\omega(1), ..., \omega(n))^t$ denotes the n states of nature where $\omega(i)$ can take one of c values $\omega_1, \omega_2, \ldots, \omega_c$ (i.e., c categories)
- Suppose $P(\boldsymbol{\omega})$ is the prior probability of the n states of nature.
- Suppose $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$ are n observed vectors.

$$P(\boldsymbol{\omega}|X) = \frac{p(X|\boldsymbol{\omega})P(\boldsymbol{\omega})}{p(X)} = \frac{p(X|\boldsymbol{\omega})P(\boldsymbol{\omega})}{\sum_{\boldsymbol{\omega}} p(X|\boldsymbol{\omega})P(\boldsymbol{\omega})}$$

$$p(X|\boldsymbol{\omega}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\omega(i))$$

It is unacceptable to simplify the problem of calculating P(ω) by assuming that the states of nature are independent.

Outline

- Bayesian Decision Theory
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- **Max Likelihood Estimation**
- Example with Real World Data

Intuition

- We could design an optimal classifier if we knew:
  - $p(\omega_i)$ (priors)
  - $p(x \mid \omega_i)$ (class conditional densities)
  - Unfortunately, we rarely have this complete information!

- Design a classifier from training data.
- Samples are often too small for class conditional estimation (large dimension of feature space)

Supervised Learning in a Nutshell

Training Stage:

- Raw Data → x
- Training Data { (x,y) } → f

(Feature Extraction)

(Learning)

Testing Stage

- Raw Data → x
- Test Data x → f(x)

(Feature Extraction)

(Apply function, Evaluate error)

Statistical Estimation View

- Probabilities to the rescue:
  - x and y are random variables
  - $D = (x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N) \sim P(X,Y)$
- IID: Independent Identically Distributed
  - Both training & testing data sampled IID from $P(X,Y)$
  - Learn on training set
  - Have some hope of generalizing to test set

Parameter Estimation

- Use a priori information about the problem

E.g.: Normality of $p(x \mid \omega_i)$

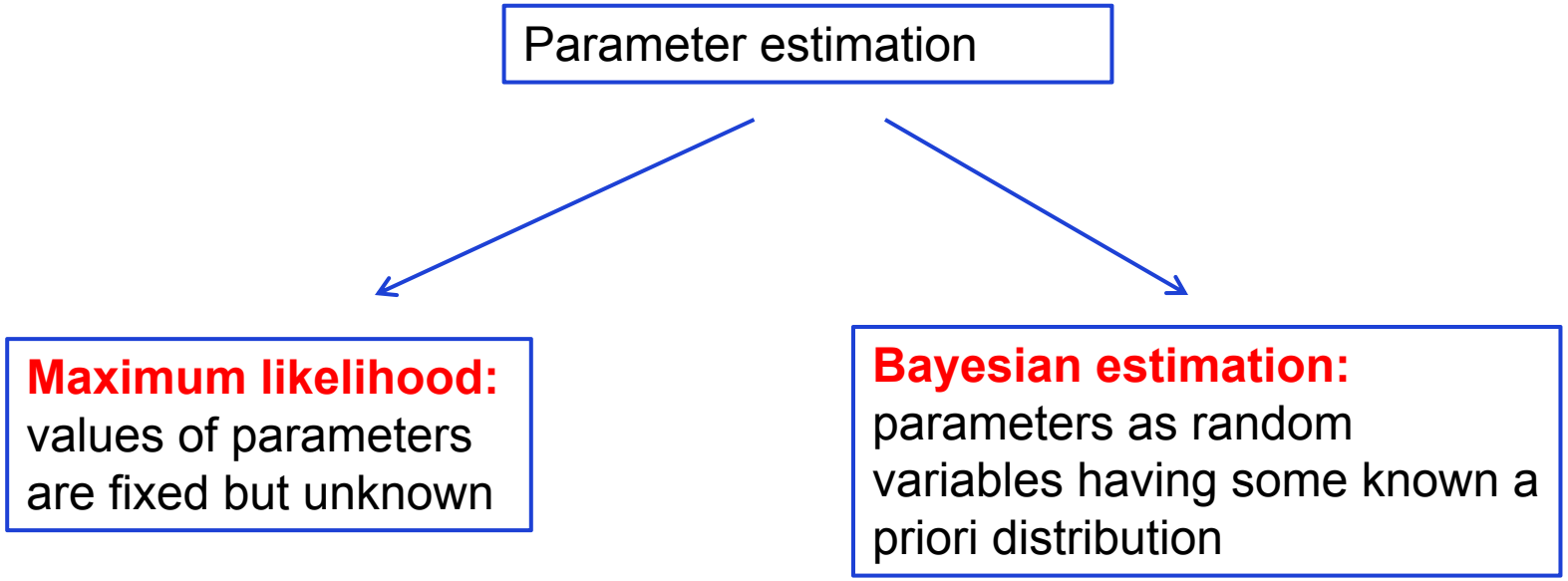
$$p(x \mid \omega_i) \sim N(\mu_i, \Sigma_i)$$

- Simplify problem
  - From estimating unknown distribution function
  - To estimating parameters

Why Gaussians?

- Why does the entire world seem to always be harping on about Gaussians?
  - Central Limit Theorem!
  - They're easy (and we like easy)
  - Closely related to squared loss (for regression)
  - Mixture of Gaussians is sufficient to approximate many distributions

Parameter Parameter



Parameter Estimation

- Parameters in ML estimation are fixed but unknown!
  - Best parameters are obtained by maximizing the probability of obtaining the samples observed.
  - Bayesian methods view the parameters as random variables having some known distribution.
  - In either approach, we use $p(\omega_i \mid x)$ for our classification rule

Maximum Likelihood Estimation: Independence Across Classes

- For each class ω_i we have a proposed density $p_i(x|\ \omega_i)$ with unknown parameters θ_i which we need to estimate.
- Since we assumed independence of data across the classes, estimation is an identical procedure for all classes.
- To simplify notation, we drop sub−indexes and say that we need to estimate parameters θ for density $p(x)$

Maximum-Likelihood Estimation

- Has good convergence properties as the sample size increases
- Simpler than alternative techniques
- General principle
  - Assume c datasets (classes) $D1, D2, \ldots, Dc$

drawn independently according to $p(x|\omega_j)$

  - Assume that $p(x|\omega_j)$ has known parametric form determined by parameter vector θ_j
  - Further assume that Di gives no information about θ_j $(i \neq j)$

Maximum-Likelihood Estimation

- Use set of independent samples to estimate $p(D \mid \theta)$

Let $D = \{x_1, x_2, \ldots, x_n\}$ $\quad |D| = n$

$$p(D \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

- Our goal is to determine $\hat{\theta}$ (value of θ that best agrees with observed training data)

- Note if D is fixed $p(D \mid \theta)$ is not a density

Example: Gaussian case

- Assume we have c classes and

$$p(x \mid \omega_j) \sim N(\mu_j, \Sigma_j)$$
$$p(x \mid \omega_j) \equiv p(x \mid \omega_j, \theta_j)$$

- Use the information provided by the training samples to estimate $\theta = (\theta_1, \theta_2, \ldots, \theta_c)$, each θ_i is associated with each category.
- Suppose that D contains n samples, $x_1, x_2, \ldots, x_n$

$$p(D \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

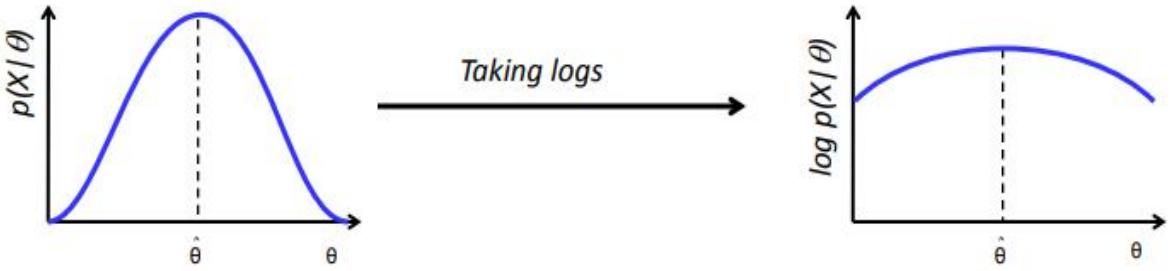
Maximum-Likelihood Estimation

$$p(D \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

- $p(D|\theta)$ is called the likelihood of θ w.r.t the set of samples.
- ML estimate of θ is, by definition the value $\hat{\theta}$ that maximizes $p(D|\theta)$

"It is the value of θ that best agrees with the actually observed training sample"

$$\hat{\theta} = argmax[p(X|\theta)] = argmax[\log p(X|\theta)]$$



Optimal Estimation

- Let $\theta = (\theta_1, \theta_2, \ldots, \theta_p)^t$ and let ∇_θ be the gradient operator

$$\nabla_\theta = \left[\frac{\partial}{\partial\theta_1}, \frac{\partial}{\partial\theta_2}, \ldots, \frac{\partial}{\partial\theta_p}\right]^t$$

- We define $l(\theta)$ as the log likelihood function

$$l(\theta) = \ln p(D \mid \theta)$$

- New problem statement:
  - determine θ that maximizes the log likelihood

$$\hat{\theta} = \arg\max_{\theta} l(\theta)$$

Optimal Estimation

$$\nabla_\theta l = \sum_{k=1}^{k=n} \nabla_\theta \ln p(x_k \mid \theta)$$

$$\nabla_\theta l = 0$$

- Local or global maximum
- Local or global minimum
- Saddle point
- Boundary of parameter space

Example of ML estimation: Unkonwn μ

- Samples are drawn from a multivariate normal population

$$\ln p(x_k \mid \mu) = -\frac{1}{2}\ln\left[(2\pi)^d |\Sigma|\right] - \frac{1}{2}(x_k - \mu)^t \Sigma^{-1}(x_k - \mu)$$

$$\nabla_\theta \ln p(x_k \mid \mu) = \Sigma^{-1}(x_k - \mu)$$

- $\theta = \mu$, Therefore
- The ML estimate μ for must satisfy:

$$\sum_{k=1}^{k=n} \Sigma^{-1}(x_k - \hat{\mu}) = 0$$

Example of ML estimation: Unkonwn μ

- • Multiplying by $\sum$ and rearranging, we obtain:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{k=n} x_k$$

- Just the arithmetic average of the samples of the training samples!

Conclusion:

If $p(x_k \mid \omega_j)$ $(j = 1, 2, \ldots, c)$ is assumed to be Gaussian in a d-dimensional feature space, then we can estimate the vector $\theta = (\theta_1, \theta_2, \ldots, \theta_c)^t$ and perform optimal classification!

Example of ML estimation: Unkonwn μ and σ^2

- Parameter: $\theta = (\theta_1, \theta_2) = (\mu, \sigma^2)$
- Objective function:

$$l = \ln\ p(x_k \mid \theta) = -\frac{1}{2}\ln 2\pi\theta_2 - \frac{1}{2\theta_2}(x_k - \theta_1)^2$$

$$\nabla_\theta l = \begin{pmatrix} \frac{\partial}{\partial \theta_1}(\ln\ p(x_k \mid \theta)) \\ \frac{\partial}{\partial \theta_2}(\ln\ p(x_k \mid \theta)) \end{pmatrix} = 0$$

$$\begin{cases} \frac{1}{\theta_2}(x_k - \theta_1) = 0 \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} = 0 \end{cases}$$

Example of ML estimation: Unkonwn μ and σ^2

- Summation

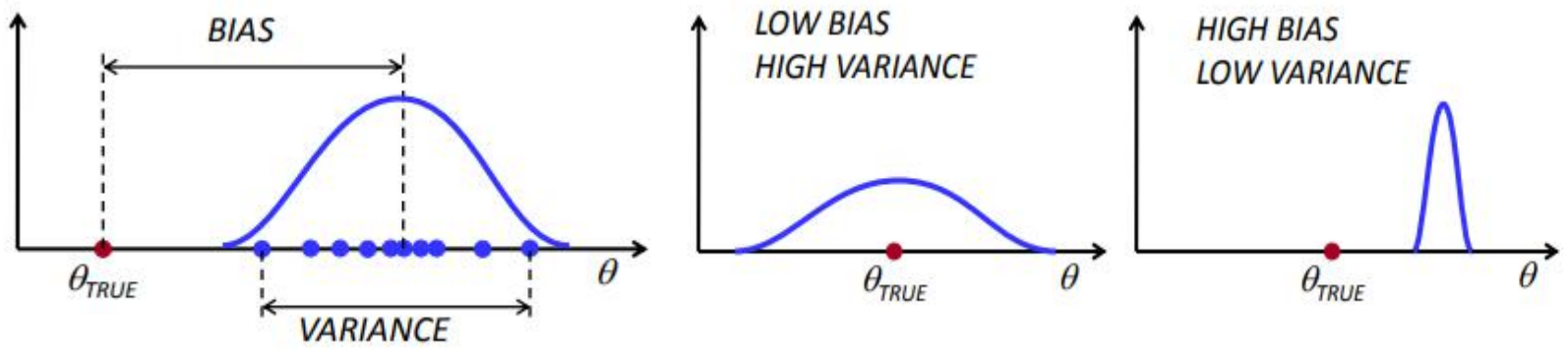
$$\begin{cases} \sum_{k=1}^{k=n} \frac{1}{\hat{\theta}_2}(x_k - \theta_1) = 0 & (1) \\ -\sum_{k=1}^{k=n} \frac{1}{\hat{\theta}_2} + \sum_{k=1}^{k=n} \frac{(x_k - \hat{\theta}_1)^2}{\hat{\theta}_2^2} = 0 & (2) \end{cases}$$

- Combining (1) and (2), one obtains:

$$\mu = \sum_{k=1}^{k=n} \frac{x_k}{n} \qquad \sigma^2 = \frac{\sum_{k=1}^{k=n} (x_k - \mu)^2}{n}$$

How good are these estimates?

- Two measures of "goodness" are used for statistical estimates
  - **BIAS**: how close is the estimate to the true value?
  - **VARIANCE**: how much does it change for different datasets?
- The bias–variance tradeoff
  - In most cases, you can only decrease one of them at the expense of the other



What is the bias of the ML estimate of the mean?

$$E[\hat{\mu}] = E\left[\frac{1}{N}\Sigma_{k=1}^{N} x^{(k}\right]$$

$$= \frac{1}{N}\Sigma_{k=1}^{N} E\left[x^{(k}\right]$$

$$= \mu$$

- Therefore the mean is an unbiased estimate.

What is the bias of the ML estimate of the variance?

$$E[\hat{\sigma}^2] = E\left[\frac{1}{N}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\mu}\right)^2\right]$$

$$= \frac{N-1}{N}\sigma^2 \neq \sigma^2$$

in the extreme case of n = 1, in which the expectation value

$$E[\cdot] = 0 \neq \sigma^2$$

- Thus, the ML estimate of variance is BIASED
  - This is because the ML estimate of variance uses $\hat{\mu}$ instead of μ
- How "bad" is this bias?
  - For $N \to \infty$ the bias becomes zero asymptotically
  - The bias is only noticeable when we have very few samples, in which case we should not be doing statistics in the first place!
- Notice that MATLAB uses an unbiased estimate of the covariance

$$\hat{\Sigma}_{UNBIAS} = \frac{1}{N-1}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\mu}\right)\left(x^{(k} - \hat{\mu}\right)^T$$

Outline

- Bayesian Decision Theory
  - Error Bound
  - ROC
  - Missing Features
  - Compound Bayesian Decision Theory
- Max Likelihood Estimation
- **Example with Real World Data**

Example with real world data (1)

- Image is acquired by the ROSIS−03 optical sensor over the University of Pavia, Italy
- Spatial dimension: 610 x 340 pixels
- Spatial resolution: 1.3m per pixel
- Spectral dimension: 103 spectral channels (0.43/ 0.86 μm)

Example with real world data (2)

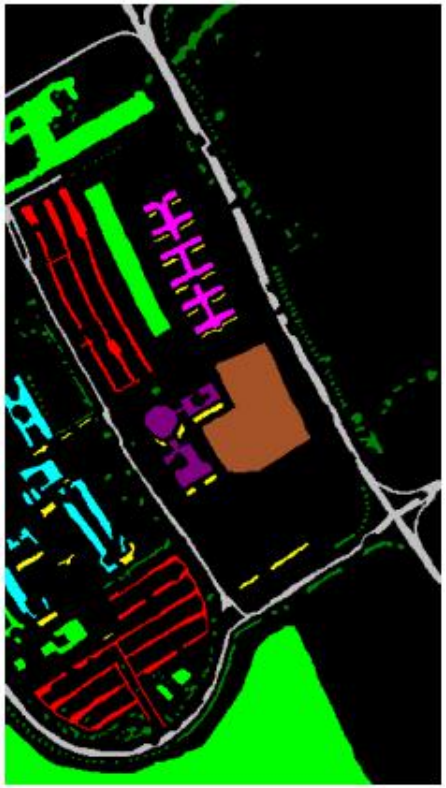
Input image (103 spectral channels)

Task:

Assign *every* pixel to *one* of the *nine* classes:

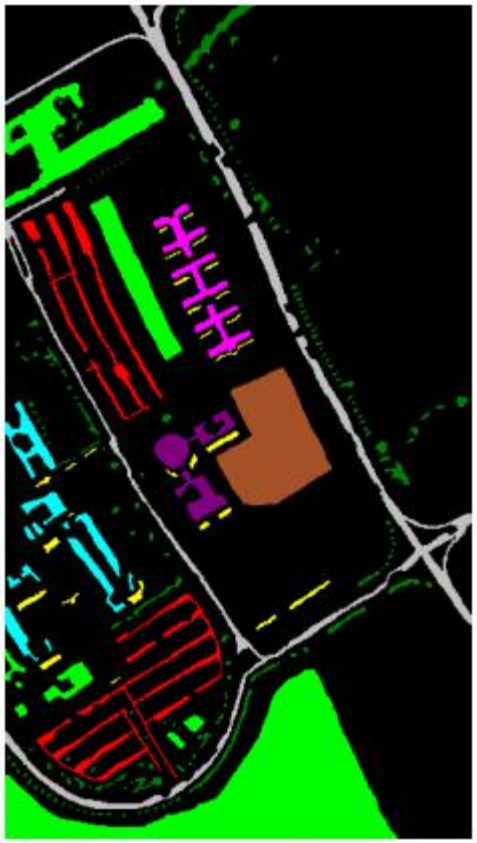
alphalt
meadows
gravel
trees
metal sheets
bare soil
bitumen
bricks
shadows

Reference data



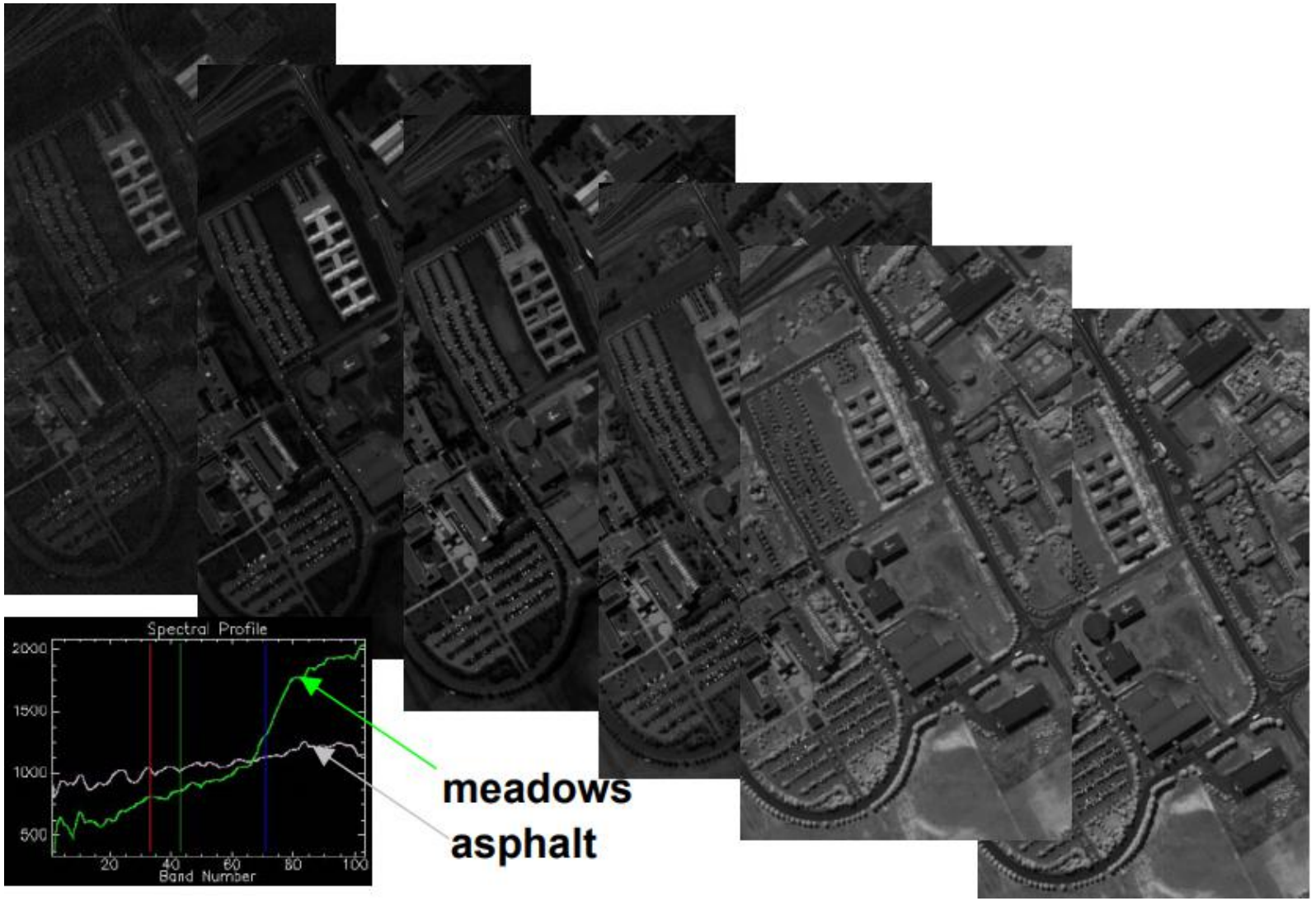
Example with real world data (3)

- We split reference data into sets of training and test samples:

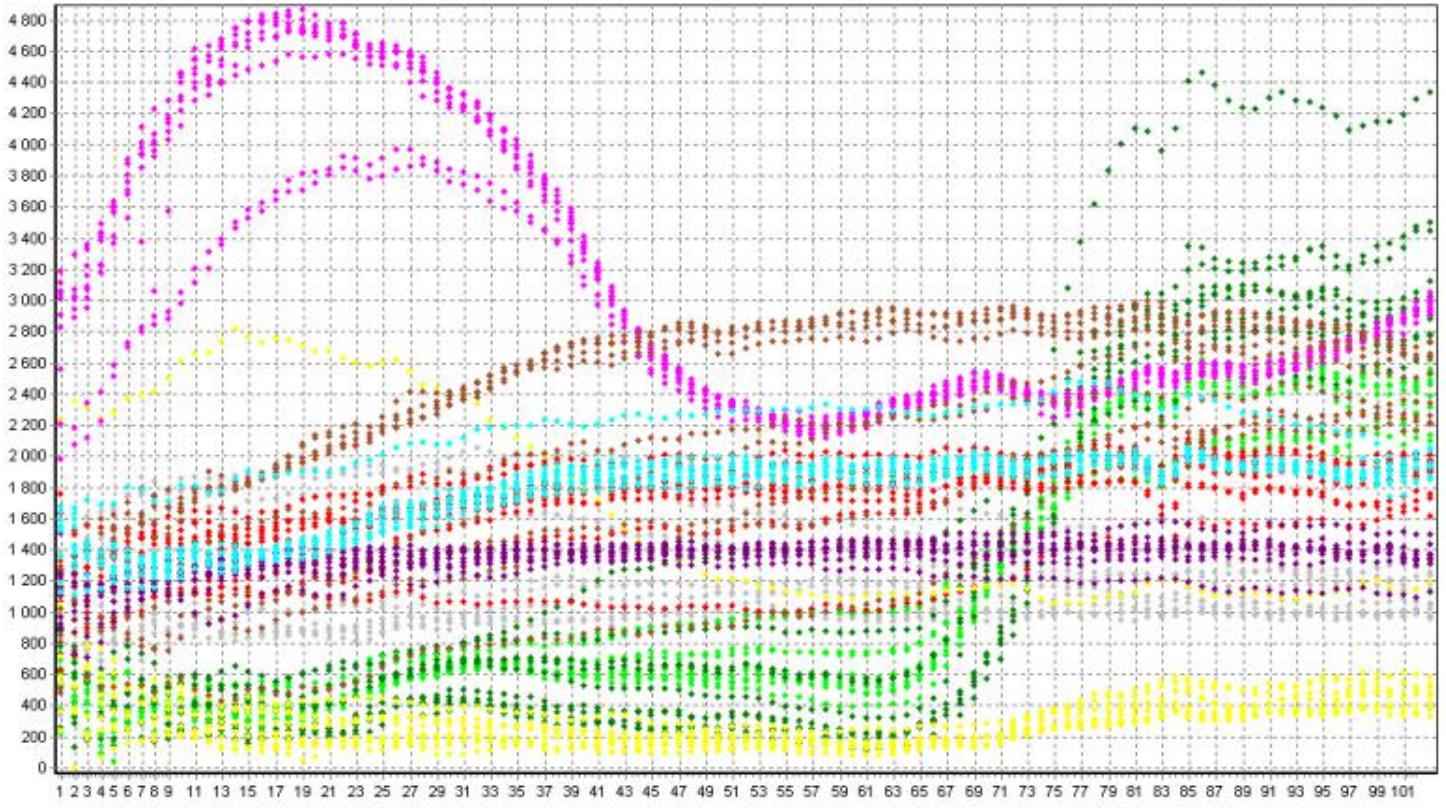


Class	Training samples	Test samples
Asphalt	**548**	**6304**
Meadows	**540**	**18146**
Gravel	**392**	**1815**
Trees	**524**	**2912**
Metal sheets	**265**	**1113**
Bare soil	**532**	**4572**
Bitumen	**375**	**981**
Bricks	**514**	**3364**
Shadows	**231**	**795**

Spectral Context for HS Image



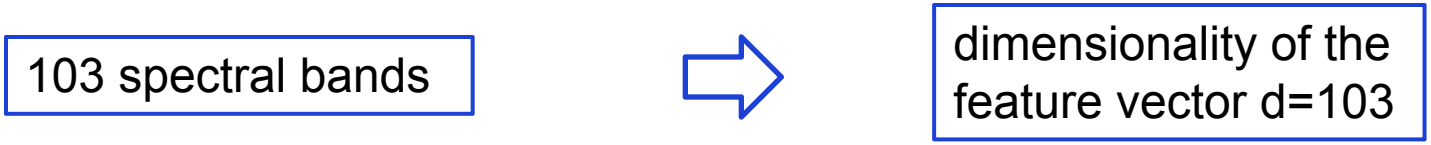
Spectral Context for HS Image



alphalt, meadows, gravel, trees, metal sheets,
bare soil, bitumen, bricks, shadows

Maximum Likelihood Classification

- Feature vector: a vector of radiance values x for each pixel



Maximum Likelihood Classification

- Samples of each class k are assumed to have a Gaussian distribution
- Parameters of distributions for each class are estimated from the training samples, using the maximum likelihood estimates:

$$\boldsymbol{\mu}_k = \frac{1}{m_k} \sum_{j=1}^{m_k} \mathbf{x}_{j,k}$$

$$\Sigma_k = \frac{1}{m_k} \sum_{j=1}^{m_k} (\mathbf{x}_{j,k} - \boldsymbol{\mu}_k)(\mathbf{x}_{j,k} - \boldsymbol{\mu}_k)^T,$$

where $\mathbf{x}_{j,k}, j = 1, ..., m_k$ - training samples for class k.

Maximum Likelihood Classification

- For each class k, $P = [d(d+1)/2 + d]$ parameters have to be estimated
- If $d = 103$, $P = 5459$!
- We have only from 231 to 548 training samples per class
- To avoid a significant parameter estimation error: P << mk (mk – number of training samples for class k)

Maximum Likelihood Classification

- Dimensionality reduction must be performed first, to reduce the dimensionality d
- The first 3 bands on the 103 band image are omitted
- A 10 band image is obtained by averaging over every 10 bands $(\text{new d} = 10)$

Maximum Likelihood Classification

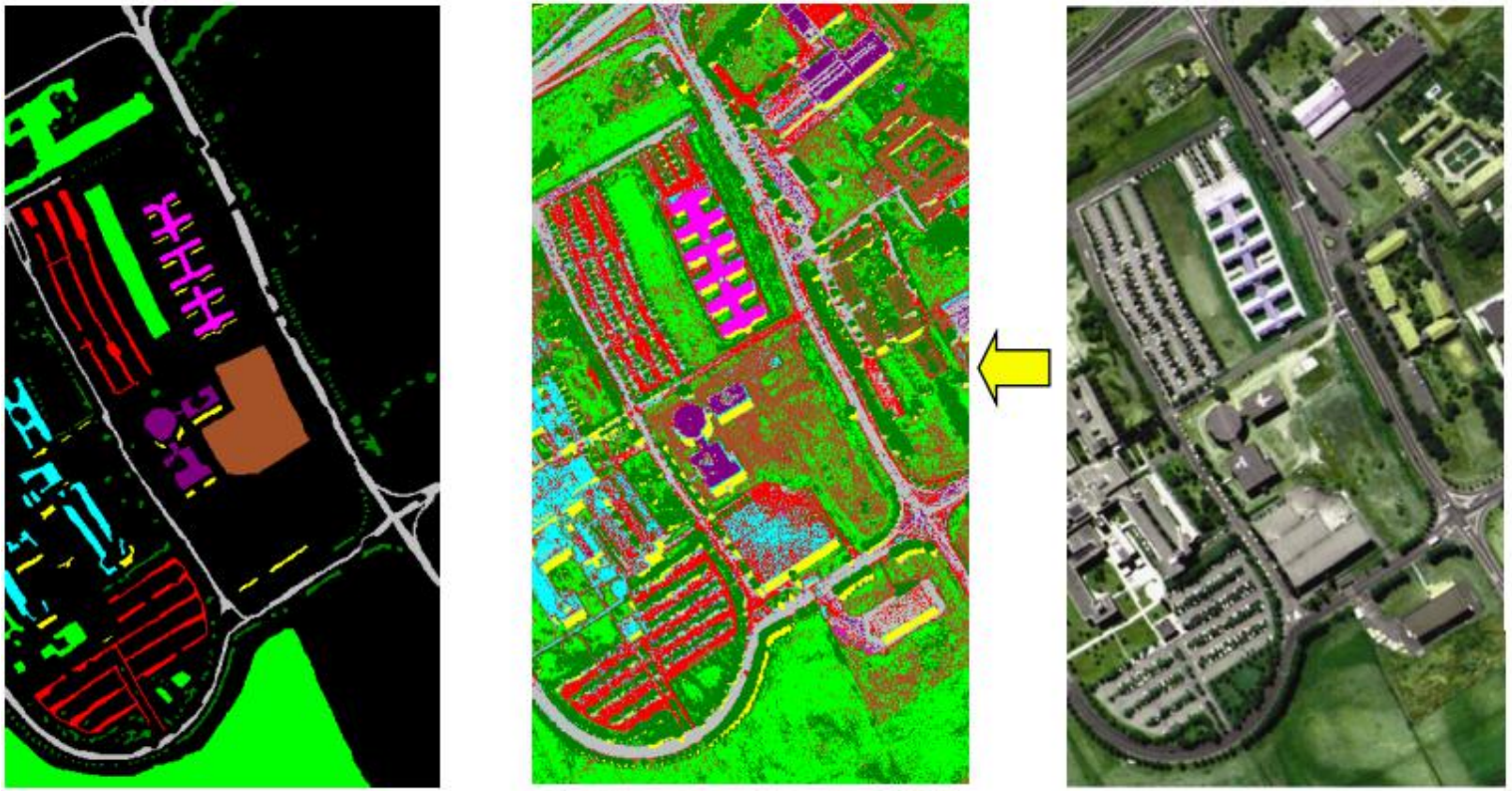
- 1) Parameters of Gaussian distributions for each class are estimated
- 2) The whole image is classified using K = 9 (number of classes) discriminant functions (MAP classification):

$$g_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(\omega_k)$$

$$P(\omega_k) = \frac{m_k}{m}$$

total number of training samples

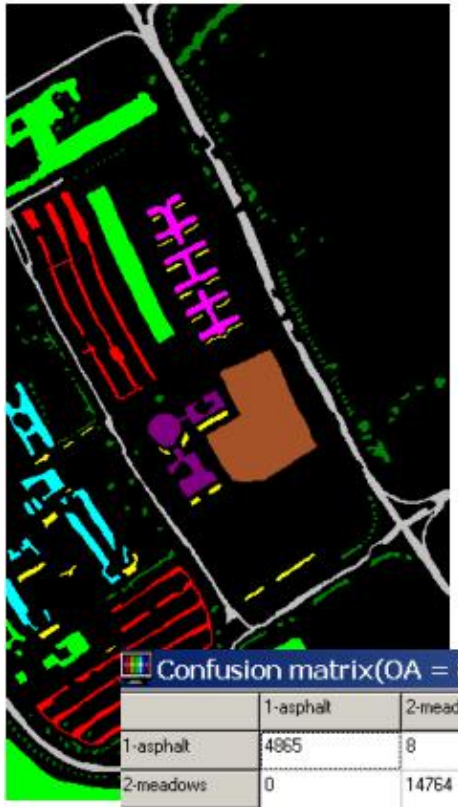
Maximum Likelihood Classification



alphalt, meadows, gravel, trees, metal sheets,
bare soil, bitumen, bricks, shadows

Overall accuracy = 82.29%

Maximum Likelihood Classification



Confusion matrix(OA = 82.289%, AA = 86.362%, K = 76.878%)

	1-asphalt	2-meadows	3-gravel	4-trees	5-metal_sheet	6-bare_soil	7-bitumen	8-brick	9-shadow	class acc.
1-asphalt	4865	8	392	39	17	34	411	538	0	77.17
2-meadows	0	14764	2	2261	0	1114	0	5	0	81.36
3-gravel	14	1	1223	1	0	8	0	568	0	67.38
4-trees	0	50	2	2858	0	3	0	0	0	98.11
5-metal_sheet	0	0	0	0	1113	0	0	0	0	100.00
6-bare_soil	1	918	68	83	0	3442	0	60	0	75.28
7-bitumen	64	0	3	1	0	1	892	20	0	90.93
8-brick	33	2	292	1	0	55	3	2978	0	88.53
9-shadow	7	0	3	0	2	0	0	0	783	98.49

Conclusions for the classification example

- Classification accuracies are high for most of the classes
- Other feature extraction (dimensionality reduction) method can be used so that accuracies can be further improved

Computational complexity

- Example: complexity of a ML estimation of the parameters in a classifier for Gaussian priors in d dimension, with n training samples
- For each of c categories

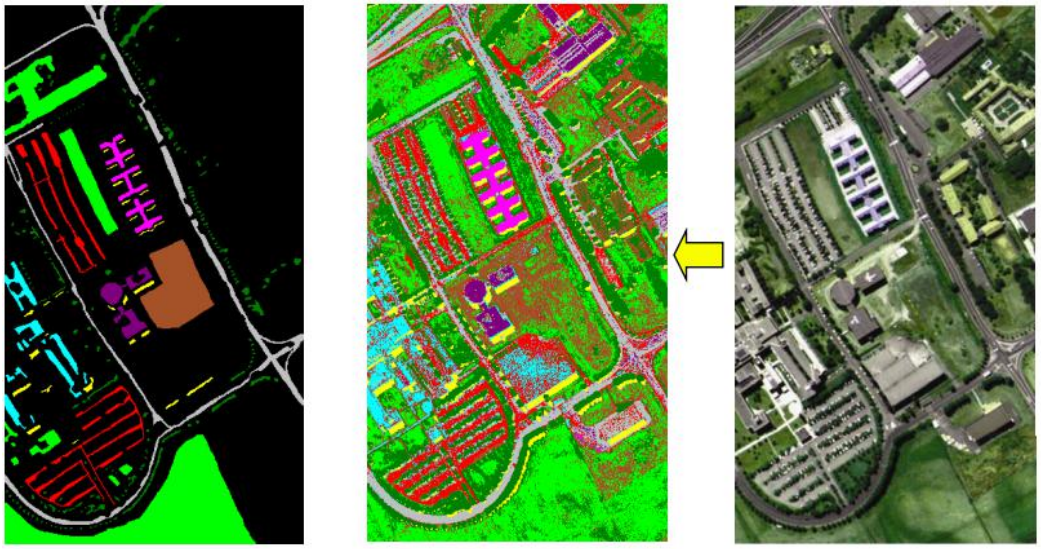
$$g(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}})^t \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\hat{\boldsymbol{\Sigma}}| + \ln P(\omega)$$

$O(dn)$, $O(nd^2)$, $O(1)$, $O(d^2n)$, $O(n)$

- Overall computational complexity for learning is ***O(cd²n)***
- Computational complexity for classificaiton of one sample is ***O(cd²)***

Computational complexity

- Parallel implementations
  - Space complexity
  - Time complexity
- Example: Estimation of the sample mean using d processors, each adding n values
  - Space complexity: O(d)
  - Time complexity: O(n)



Q & A