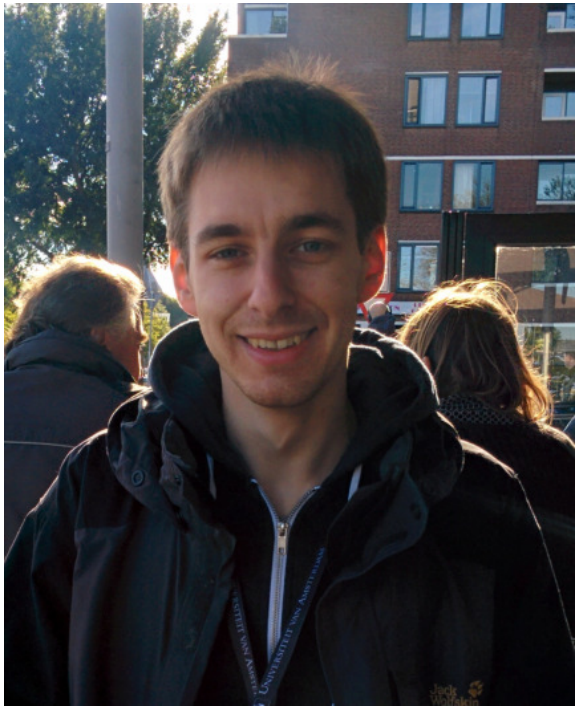
GeoX Young Academy: Machine Learning in Remote Sensing Best practice and recent developments - Part 1: Introduction to Machine Learning

Ronny Hänsch & Andreas Ley
Dept. Computer Vision and Remote Sensing
TU Berlin - Germany

Lecturers



Andreas Ley
andreas.ley@tu-berlin.de

Ronny Hänsch
r.haensch@tu-berlin.de

Goals of the tutorial

- Learn and revise the most important concepts of machine learning
- Give tips for the use of machine learning in real-life remote sensing applications
- Introduce modern learning methods

References

- Duda, Hart & Stork, "Pattern classification," 2000
- Bishop, "Neural networks for pattern recognition," 1995
- IEEE TGRS, **https://arxiv.org/**

Tutorial schedule

- 09:00 - 10:30: Part I - Basics
  - Naive Bayes, maximum likelihood, features, performance evaluation
  - Exercise: PCA
- 11:00 - 12:30: Part II - Classical learning
  - Perceptron, kNN, Support Vector Machines
  - Exercise: kNN
  - Exercise: SVM
- 13:30 - 15:00: Part III - Advanced learning
  - Random Forests, MRF, Artificial Neural Networks
- 15:30 - 17:00: Part IV - Modern learning
  - Deep learning
  - Exercise: Deep learning

What is machine learning?

What is machine learning?

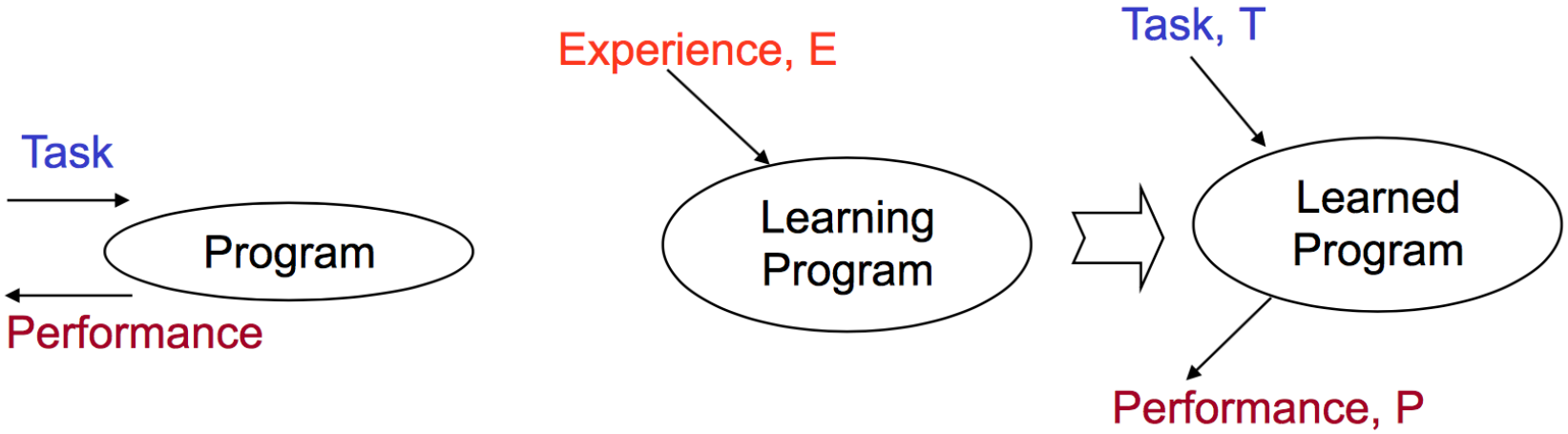
- Term introduced by Arthur Samuel in 1959:
  - Machine learning gives computers the ability to learn without being explicitly programmed

What is machine learning?

- Term introduced by Arthur Samuel in 1959:
  - Machine learning gives computers the ability to learn without being explicitly programmed
- Evolved from the study of pattern recognition and computational learning theory
- Machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning formal definition

- A computer program is said to **learn** from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its performance at tasks in **T**, as measured by **P**, improves with **experience E** [Tom M. Mitchell]



Main categories of tasks

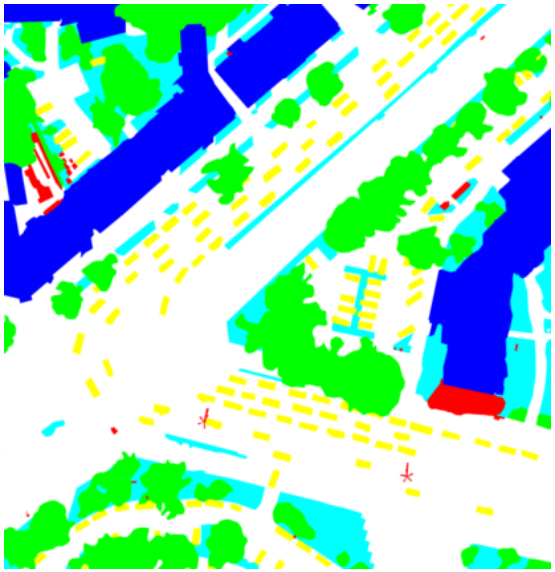
- **Supervised learning**:
  - The computer is presented with example inputs and desired outputs = **training data** $D = \{(\mathbf{x}, y)\}$
  - The goal is to learn a general rule f_w that maps inputs to outputs $f_w(\mathbf{x}) = y$
  - **Classification**: y is a nominal number (i.e. a class label)
  - **Regression**: y is a continuous number

Remote sensing image classification

Classification: assign a semantic class to every pixel

Input

$\Rightarrow$



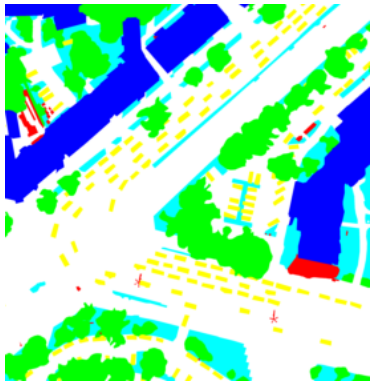
Output

- Impervious surf.
- Building
- Low veget.
- Tree
- Car
- Clutter

Remote sensing image classification

Remote sensing

$\Rightarrow$



Classification

$\Rightarrow$ "Airfield"

Land use classification

Computer vision

$\Rightarrow$ "Airplane"

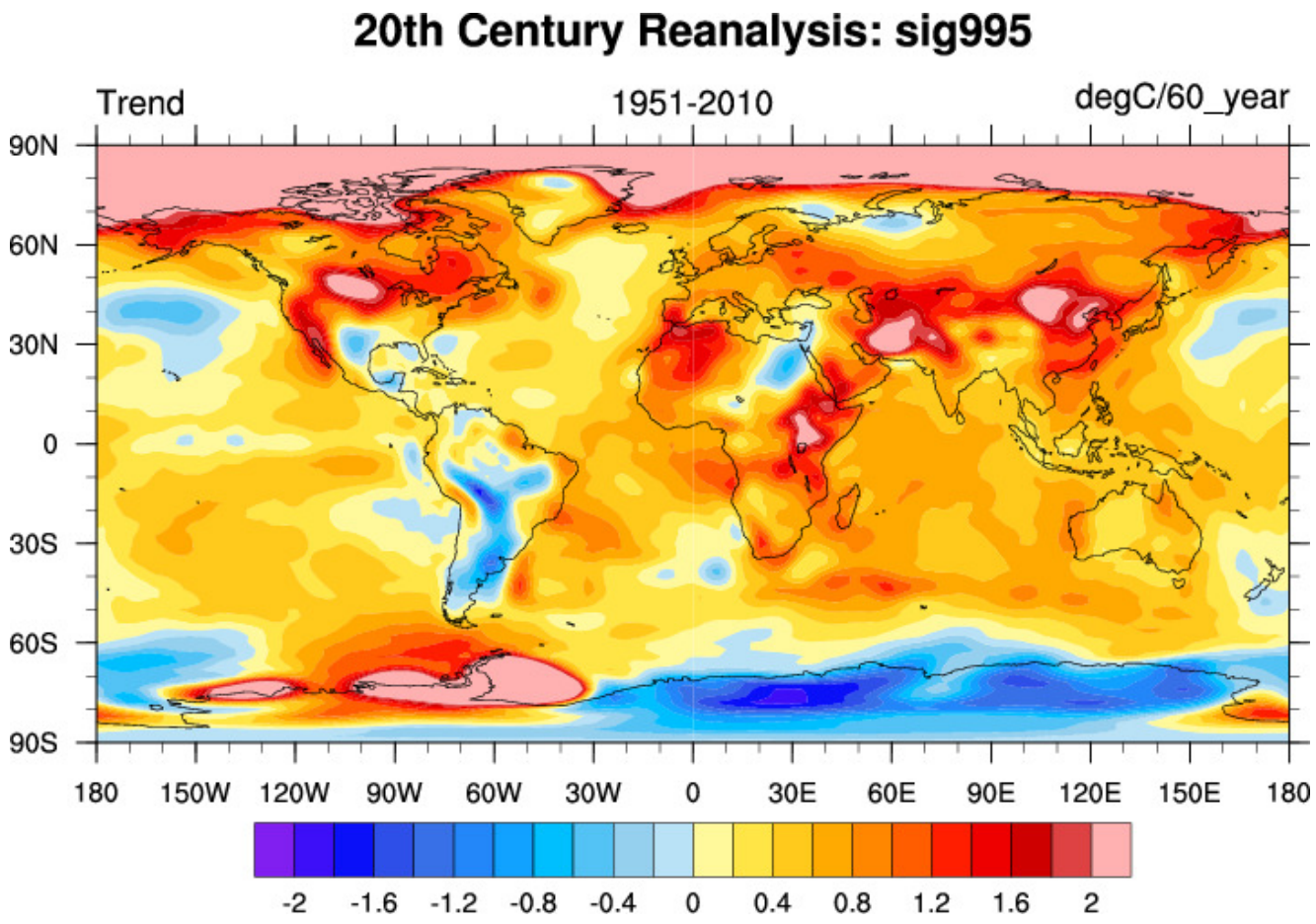
Classification

$\Rightarrow$

Semantic segmentation,
dense labeling

Regression in remote sensing

- **Example of regression:** map of annual mean temperatures

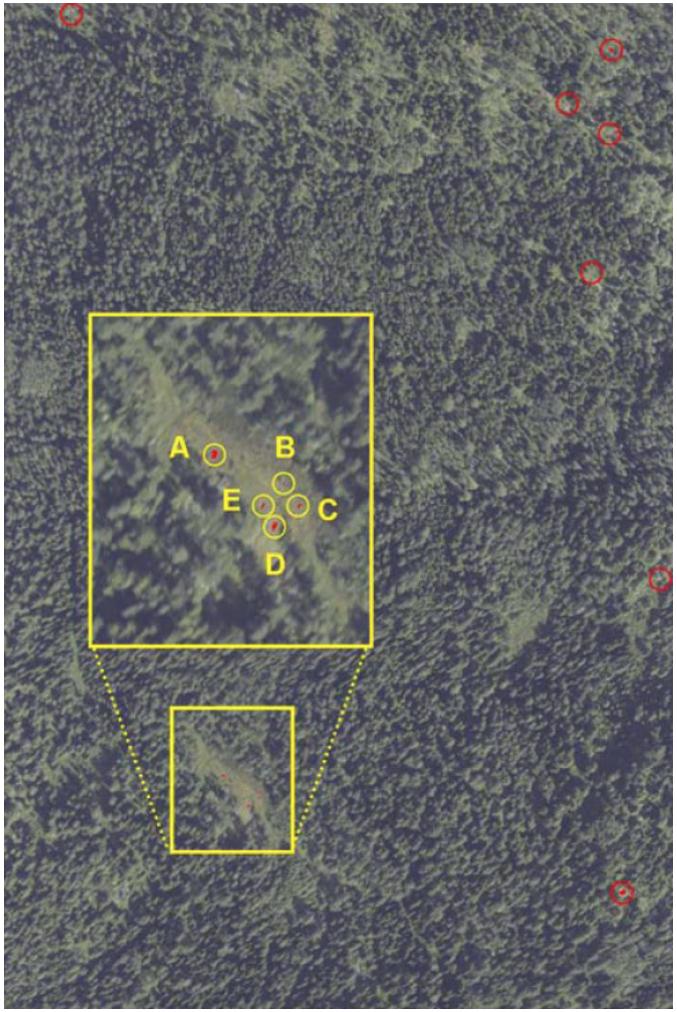


Main categories of tasks

- **Unsupervised learning**:
  - No labels are given to the learning algorithm
  - Goal: find structure in its input
  - Used to discover hidden patterns in data, learn features
  - **Clustering**

Unsupervised learning in remote sensing

- Example: **Anomaly detection**:
  - **Estimate** a background model by fitting a multivariate normal mixture model to a spatial subset of the image
  - **Evaluate** a background probability value for all pixels in the image based on the estimated model
  - **Detect** anomalous pixels by thresholding on low background probability values



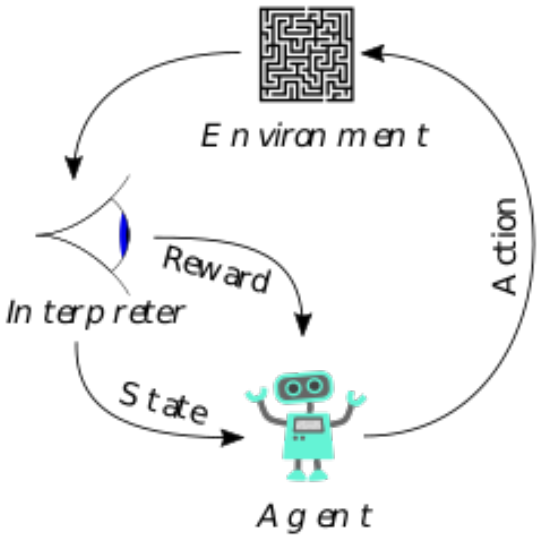
Y. Tarabalka et al., "Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing," JRTIP, 2009.

Main categories of tasks

- **Domain adaptation**:
  - We aim at learning from a source data distribution a well performing model on a different (but related) target data distribution
  - Example: images taken on different dates, with different sensors
  - Useful for large-scale classification

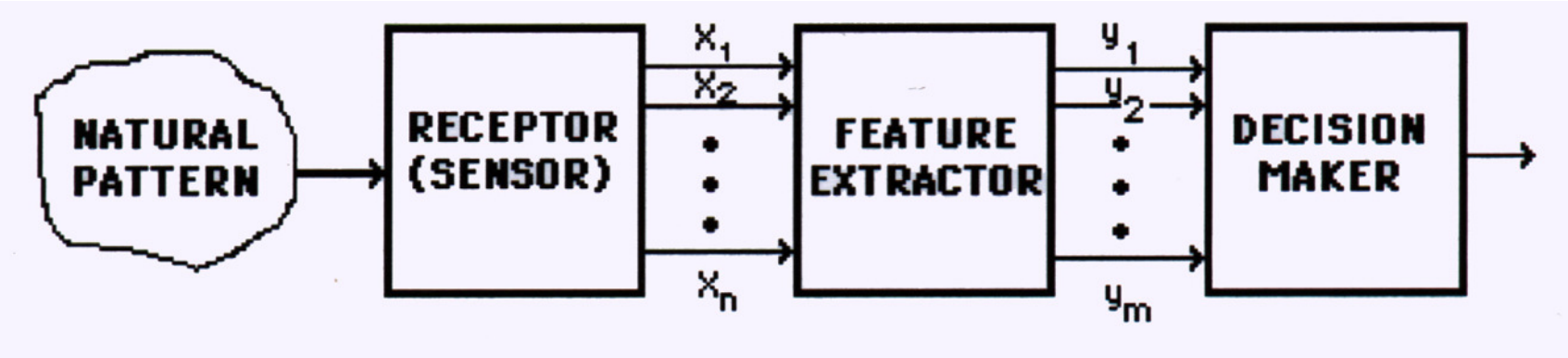
Main categories of tasks

- **Reinforcement learning**:
  - A computer program interacts with a dynamic environment in which it must perform a certain goal
  - The program is provided feedback in terms of rewards and punishments as it navigates its problem space
  - Example of goals: driving a vehicle
  - Not popular in remote sensing up to now



Classical pattern recognition system

- Pattern recognition is the science of making inferences from perceptual data, using tools from statistics, probability, computational geometry, machine learning, signal processing, and algorithm design



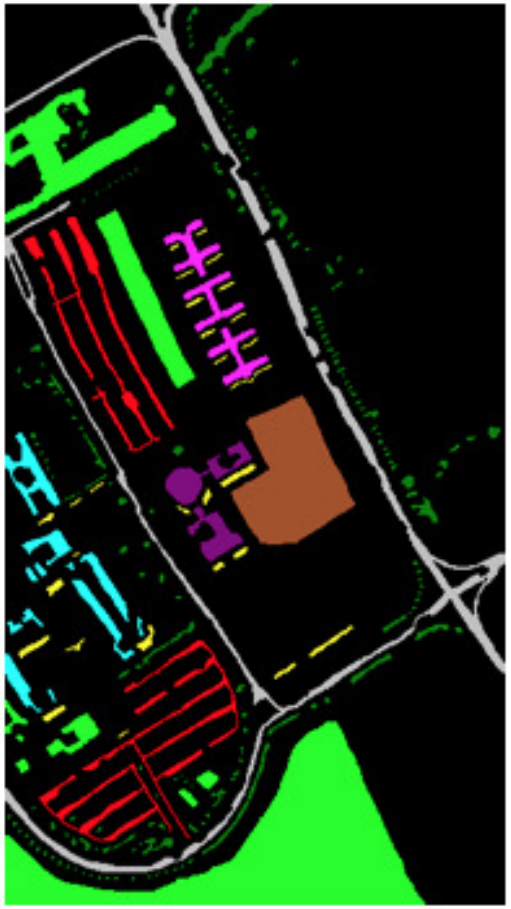
Example: hyperspectral image classification

Input image (103 spectral channels)

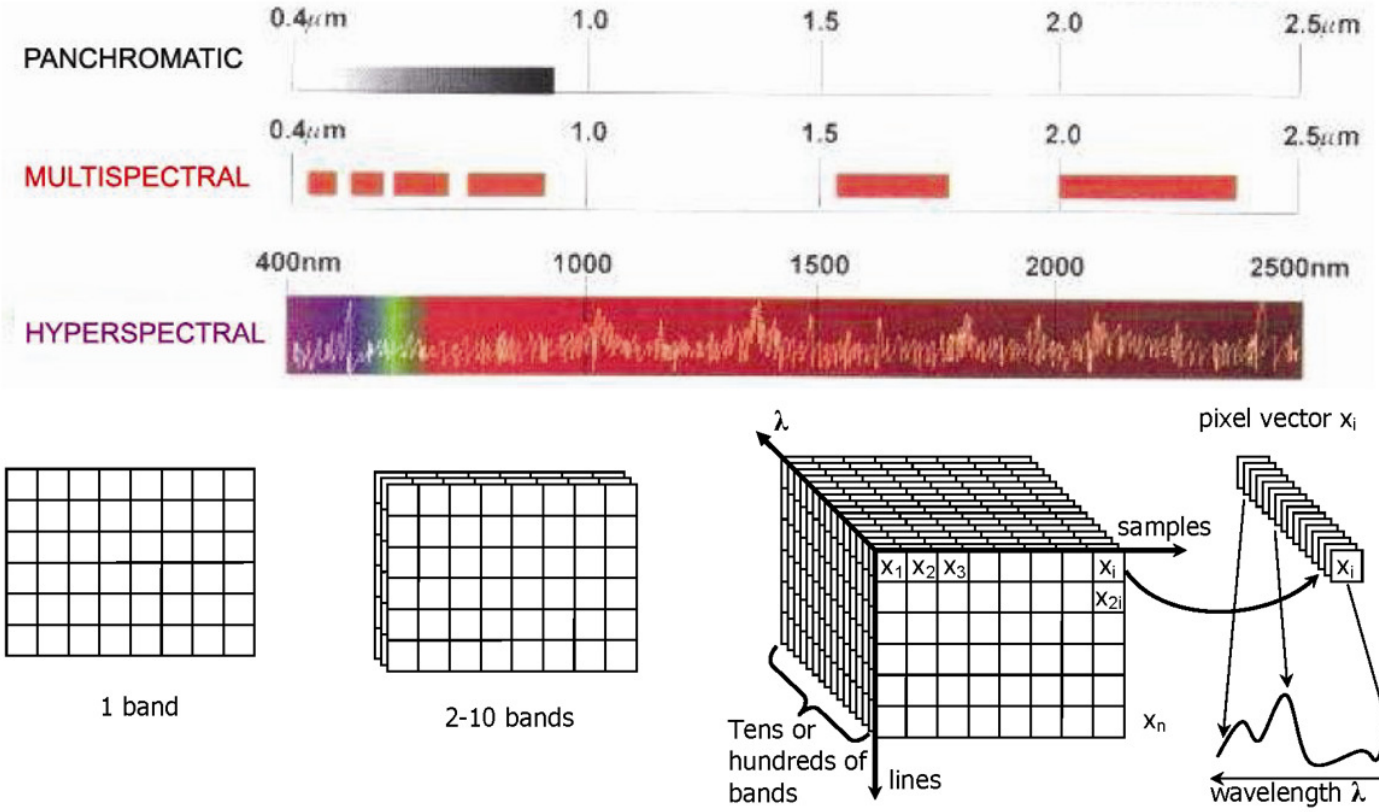
Task: assign every pixel to one of the nine classes

alphalt
meadows
gravel
trees
metal sheets
bare soil
bitumen
bricks
shadows

Reference data



Spectral context

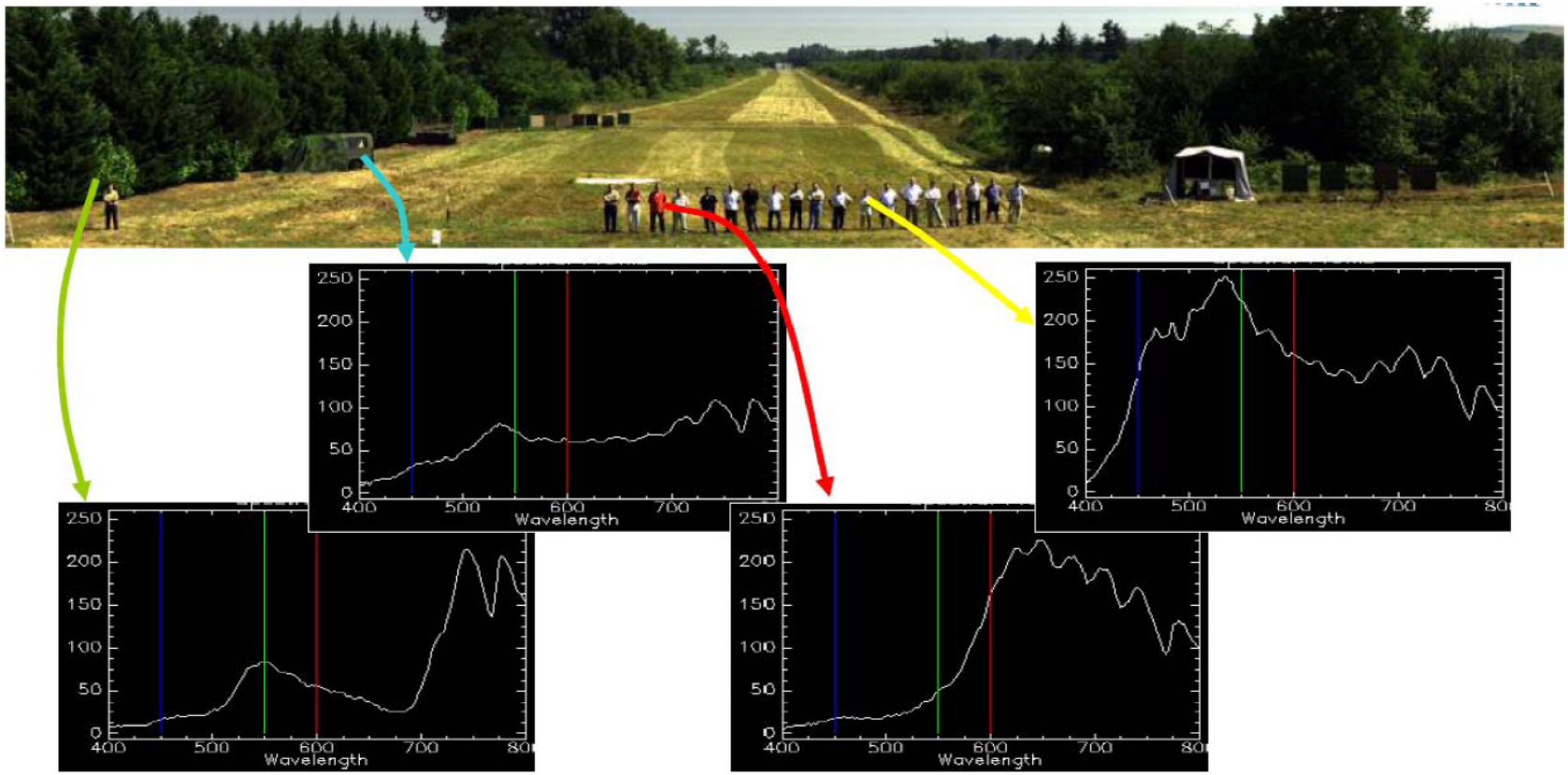


Panchromatic: one grey level value per pixel

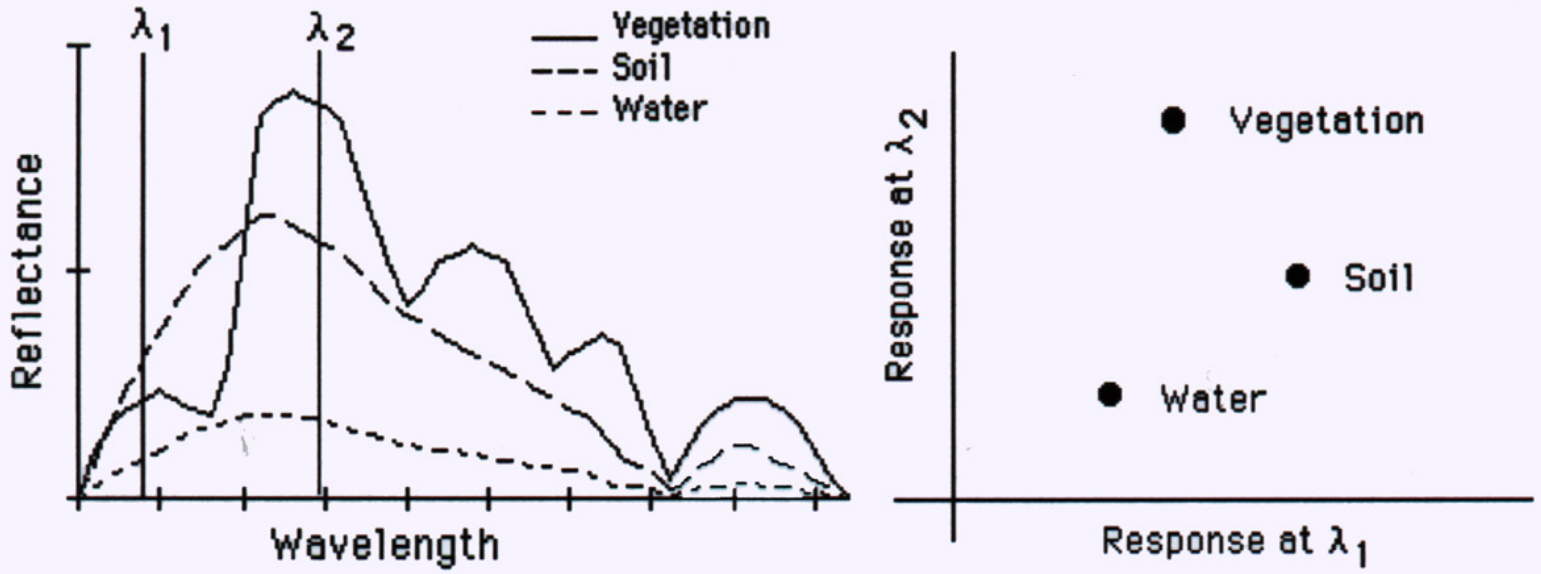
Multispectral: limited spectral info

Hyperspectral: detailed spectral info

Spectral context

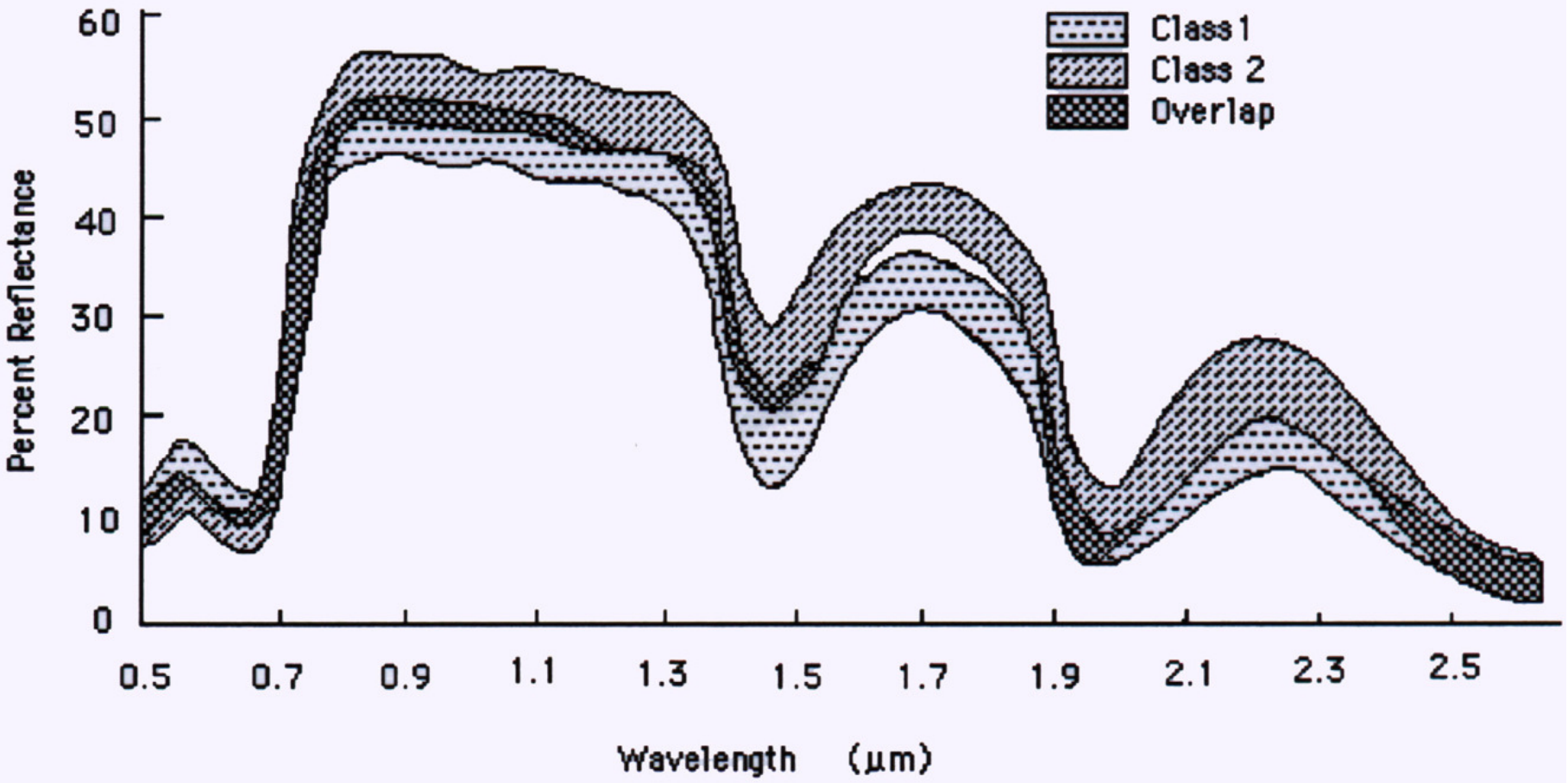


Feature space



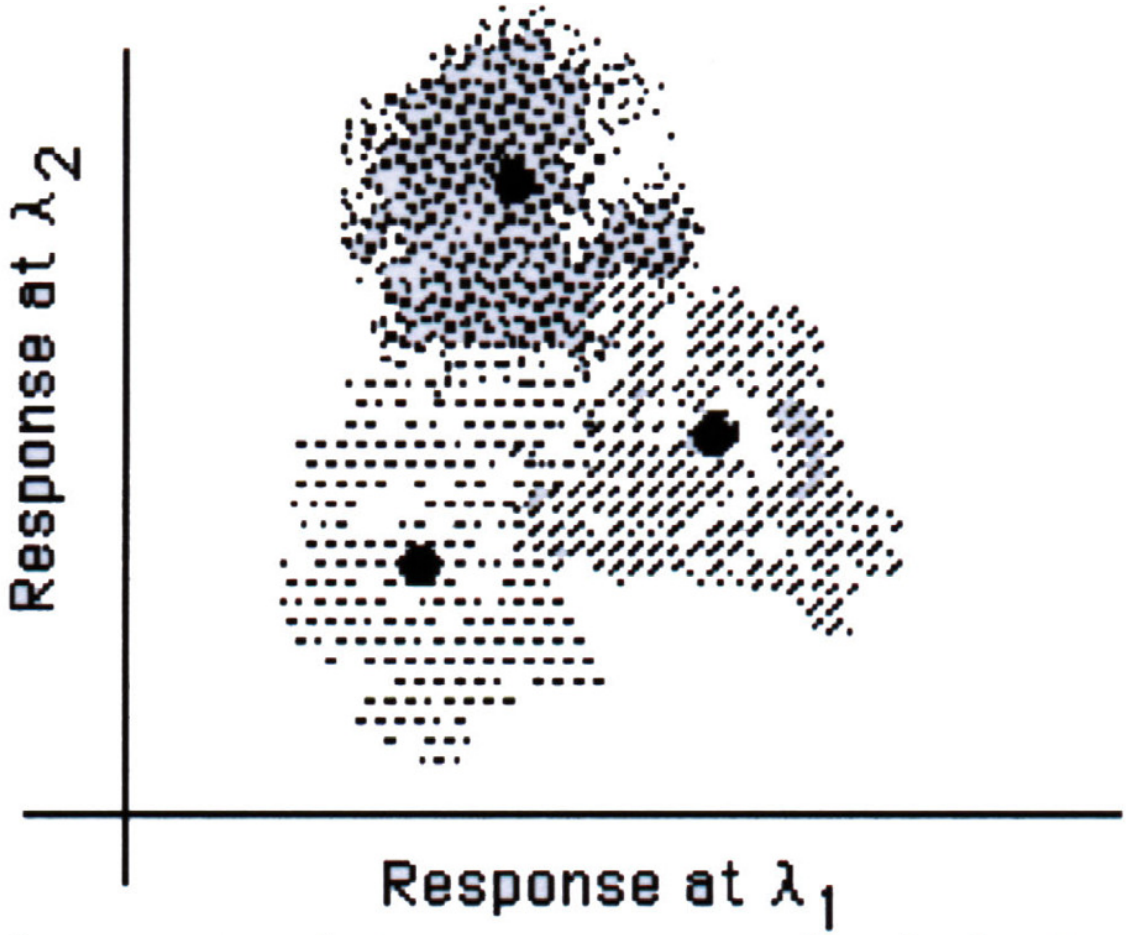
Three spectral responses mapped to a multivariant feature space. Though sampling at only two wavelengths is shown here, sampling at any number of wavelengths could be used, resulting in a correspondingly higher dimensional feature space.

There is no such thing as a clean pixel



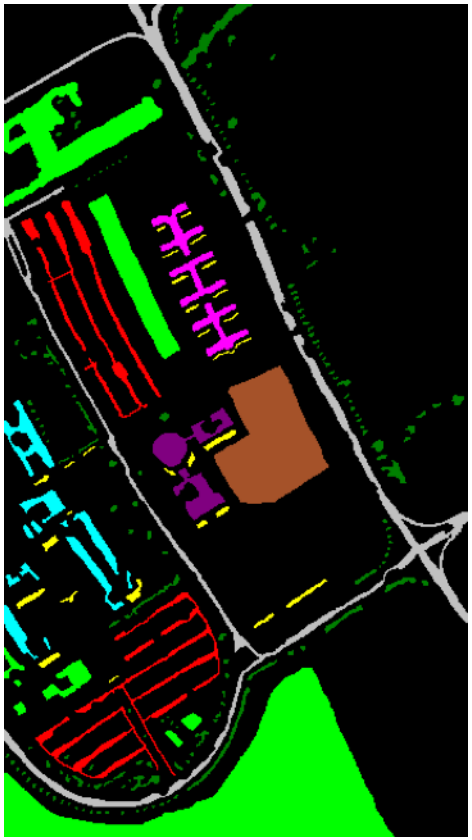
Spectral responses for a typical pair of classes, showing the interval at each wavelength into which they fall.

Feature space



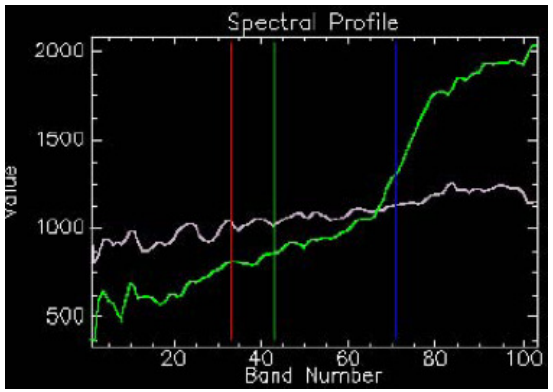
Data for classes tends to occupy a region in feature space rather than occurring at a single point.

Support vector machines classification



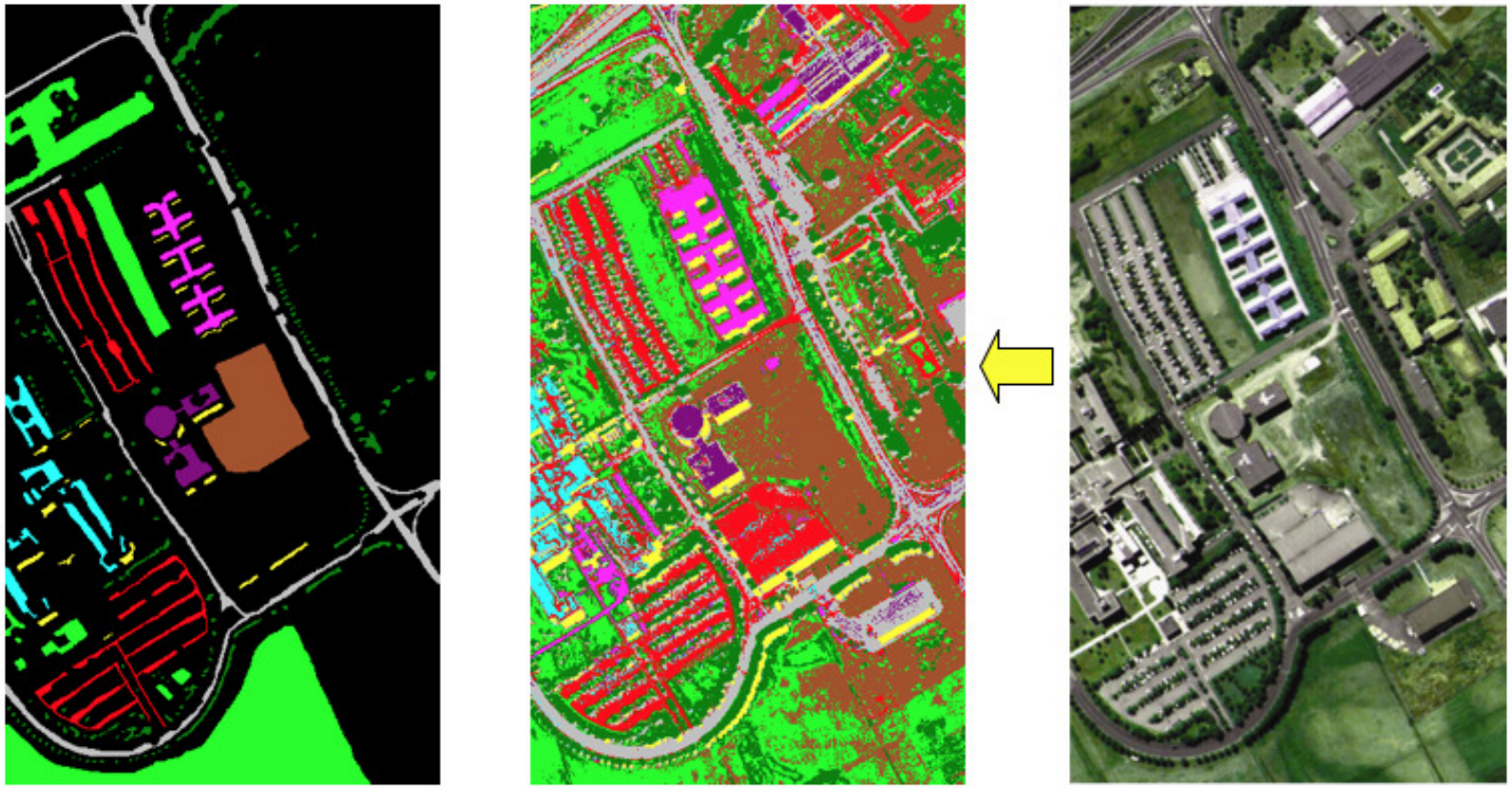
Class	Training samples	Test samples
Asphalt	548	6304
Meadows	540	18146
Gravel	392	1815
Trees	524	2912
Metal sheets	265	1113
Bare soil	532	4572
Bitumen	375	981
Bricks	514	3364
Shadows	231	795

- Spectrum of each pixel is analyzed



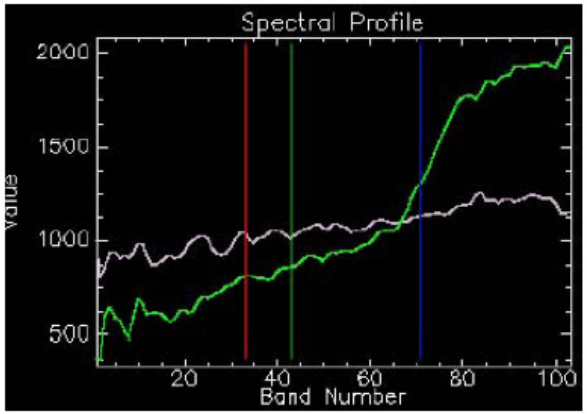
Support vector machines classification

- Overall accuracy = 81%

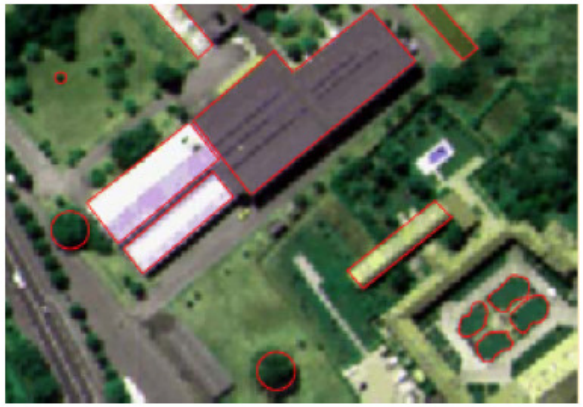


alphalt, meadows, gravel, trees, metal sheets,
bare soil, bitumen, bricks, shadows

Spectral-spatial classification



+



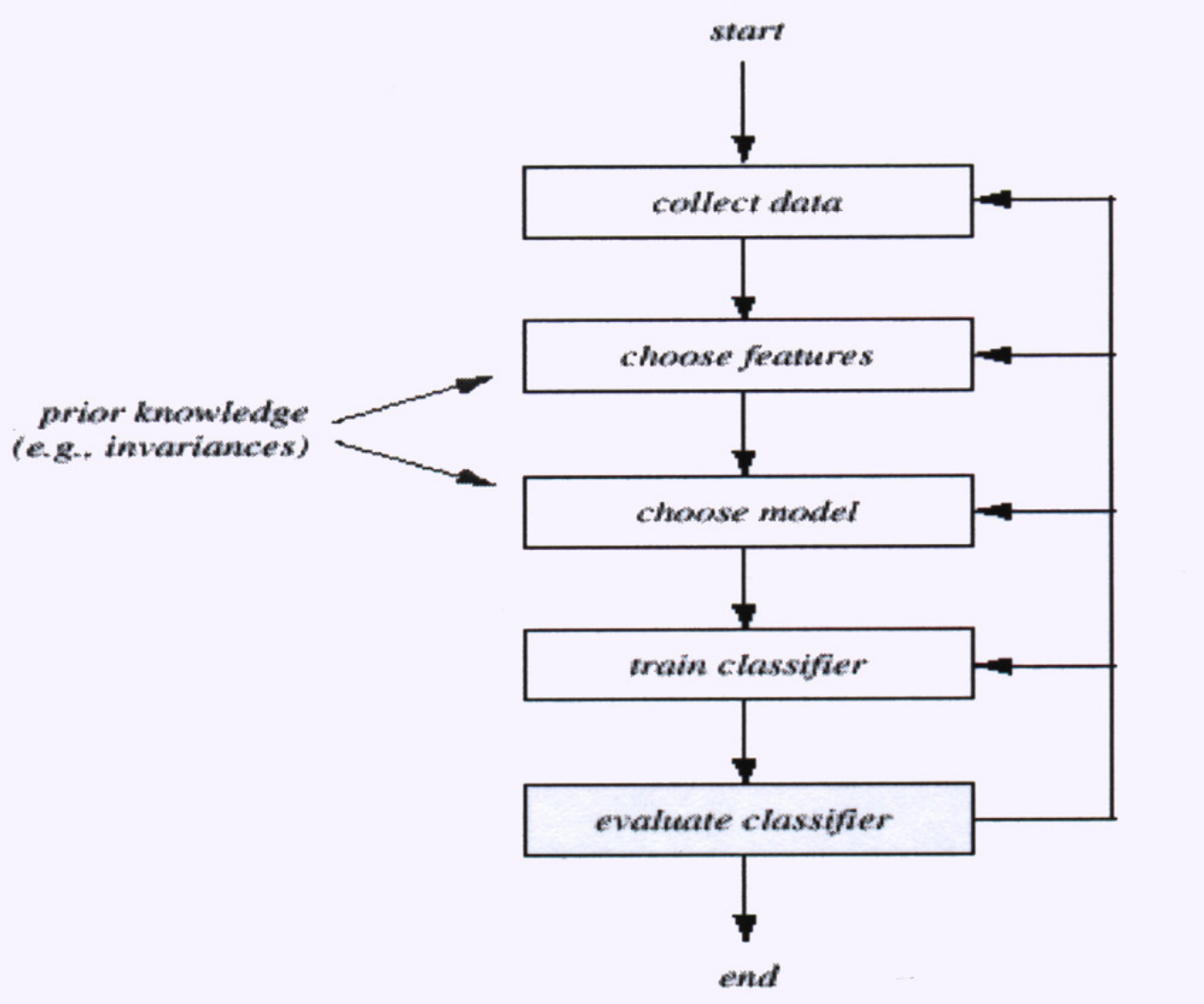
- Info about spatial structures included
  - For more accurate classification
- How to define/use info about spatial structures automatically?

Spectral-spatial classification

- Overall accuracy = 94%

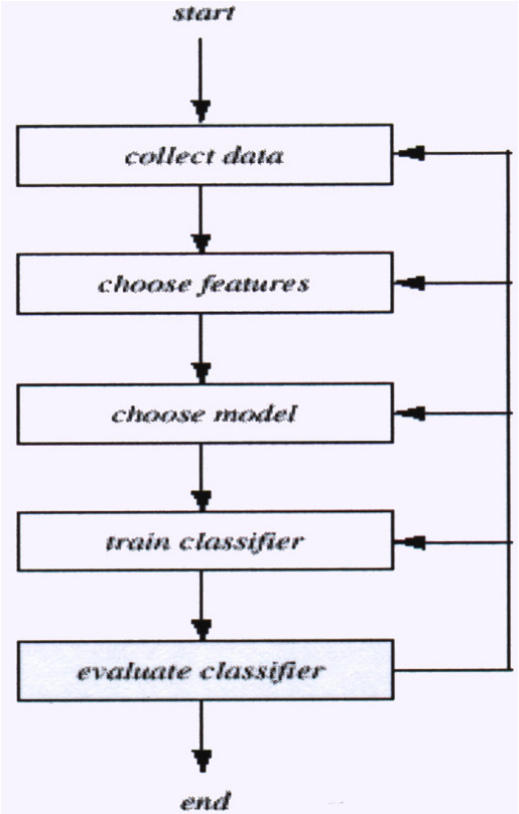
alphalt, meadows, gravel, trees, metal sheets, bare soil, bitumen, bricks, shadows

Classical design cycle



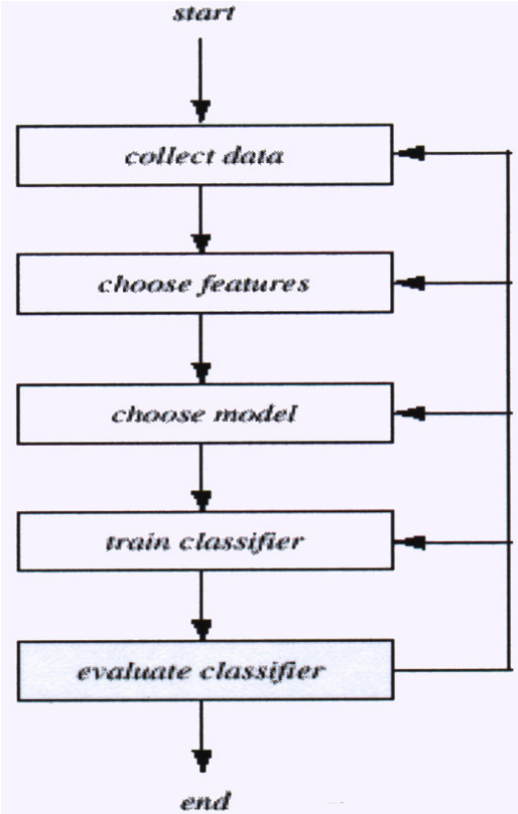
Classical design cycle

- Data collection
  - How do we know when we have collected an adequately large & representative set of examples to train/test the system?



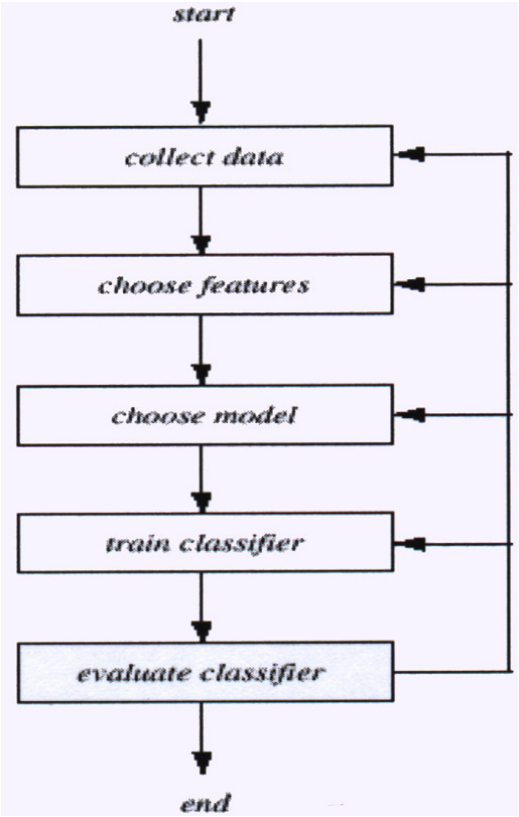
Classical design cycle

- Data collection
  - How do we know when we have collected an adequately large & representative set of examples to train/test the system?
- Feature choice
  - Depends on the characteristics of the problem domain
  - Preferably:
    - Simple to extract
    - Insensitive to noise
    - Useful to discriminate patterns in different categories
    - ...



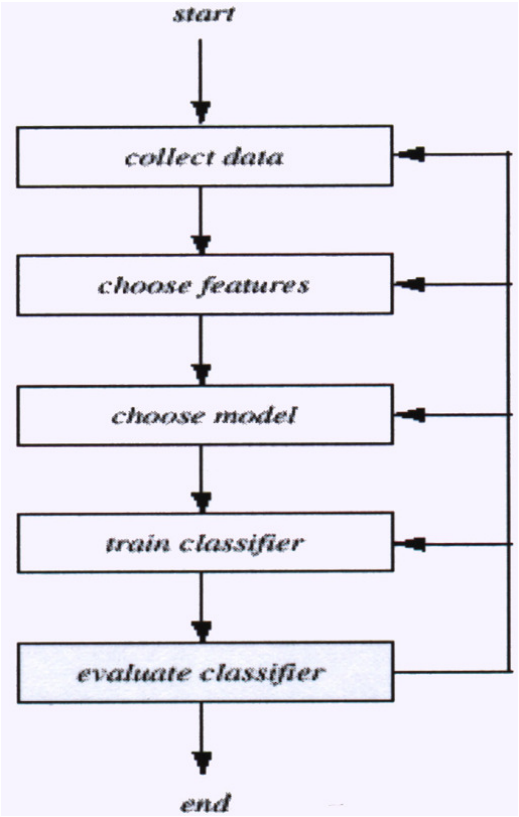
Classical design cycle

- Model choice
  - Depends on the characteristics of the problem domain
  - Often useful to test several models
- Training
  - Training set must be representative
  - Importance of cross validation / separate datasets



Classical design cycle

- Evaluation
  - Measure the error rate (or performance)
  - Test different set of features/models to compare performace
- Computational Complexity
  - What is the trade off between computational ease and performance?
  - How an algorithm scales as a function of the number of features or categories?



Modern trend

Deep learning:

- Automatically learn features if a lot of training data are available

Bayesian decision theory

- Fundamental statistical approach to the problem of pattern classification
- Assumptions:
  1. Decision problem is posed in probabilistic terms
  2. Ideal case: probability structure underlying the categories is known perfectly

Statistical decision theory

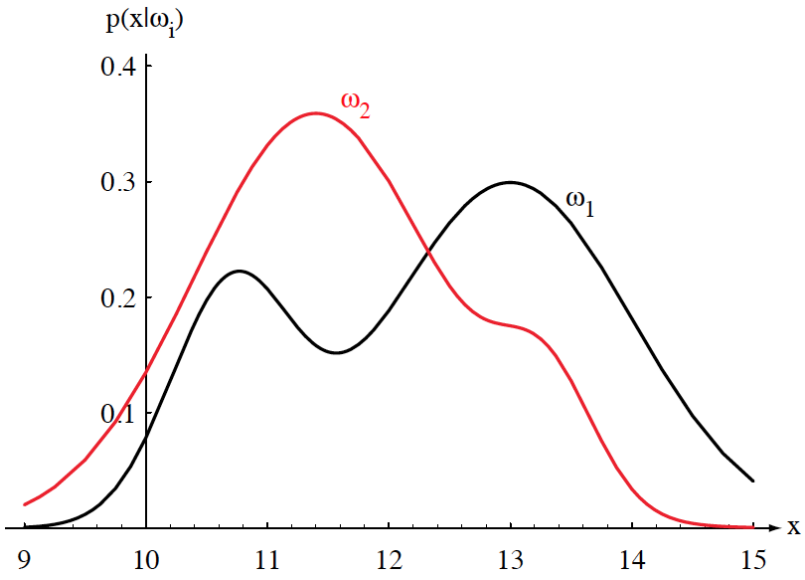
- Fish classification example:
  - Sort incoming fish on a conveyor according to species using optical sensing
  - 2 species: sea bass & salmon
- State of nature is unpredictable
  - We denote state of nature by a random variable ω
  - $\omega = \omega_1$ for sea bass; $\omega = \omega_2$ for salmon
- Prior probability
  - Reflect our prior knowledge of how likely the next fish is sea bass/salmon
  - $P(\omega_1) = P(\omega_2) \rightarrow$ the catch of sea bass/salmon is equiprobable

Statistical decision theory

- Exclusivity and exhaustivity:
  - If only two species, $P(\omega_1) + P(\omega_2) = 1$
- Decision rule with only the prior information
  - Decide ω_1 if $P(\omega_1) > P(\omega_2)$
  - Otherwise, decide ω_2
- Impossible to make good classifier with so little information

Statistical decision theory

- Use class-conditional information
- Example: we might use a lightness measurement x to improve our classifier
- $p(x|\omega_1)$ and $p(x|\omega_2)$ are **class-conditional probability density** functions
  - Describe the difference in lightness between populations of sea bass and salmon



Bayes' formula

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

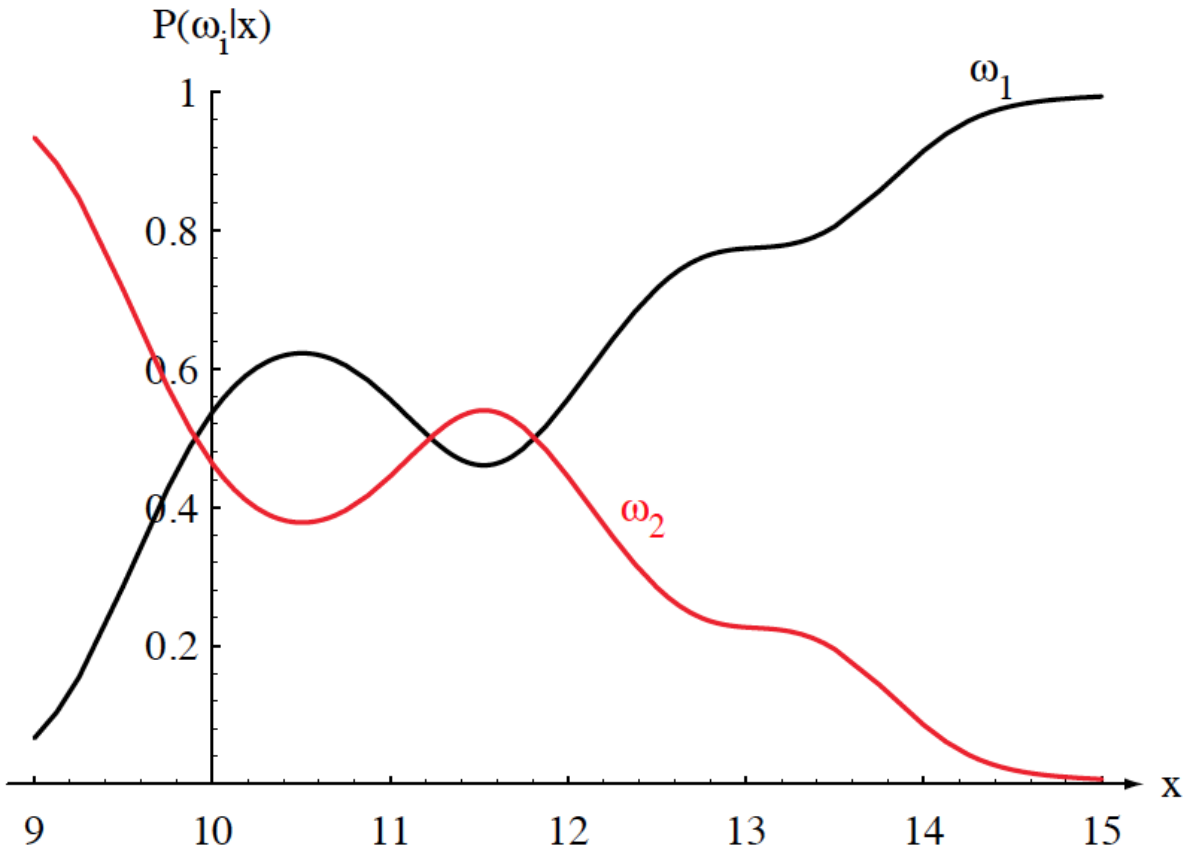
- Where in case of two categories:

$$p(x) = \sum_{j=1}^{2} p(x|\omega_j)P(\omega_j)$$

- $p(x)$ can be seen as a scale factor, guaranteeing that the posterior probabilitiess sum to 1
- Bayes' formula in words:

$$posterior = \frac{likelihood \times prior}{evidence}$$

Posterior probabilities - example



- At every x, the posteriors sum to 1

Decision given the posterior probabilities

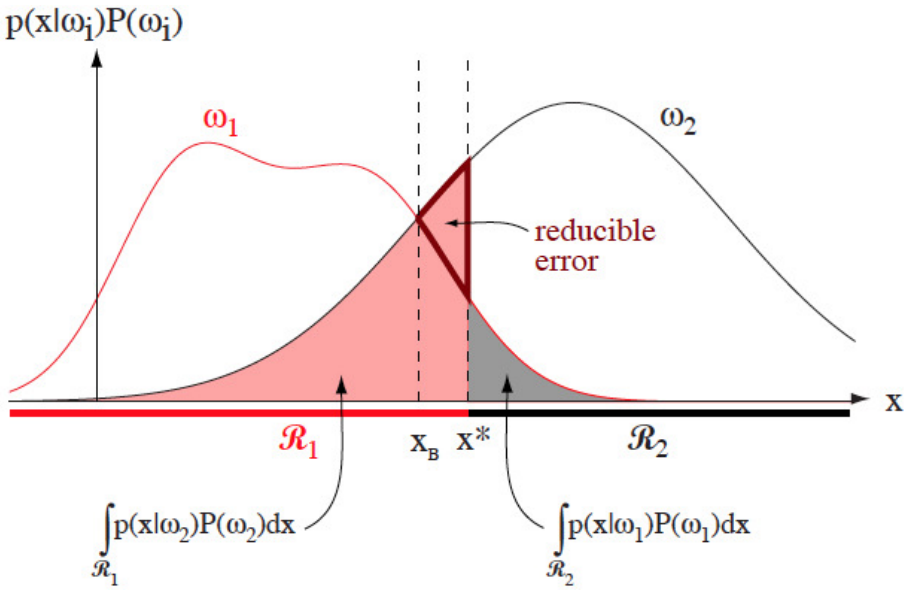
- Given an observation x:
  - If $P(\omega_1|x) > P(\omega_2|x) \Rightarrow$ we will choose ω_1
  - If $P(\omega_2|x) > P(\omega_1|x) \Rightarrow$ we will choose ω_2
- Therefore, whenever we observe a particular x, the probability of error:

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

Bayes decision rule

- Minimizing the probability of error:
  - Decide ω_1 if $P(\omega_1|x) > P(\omega_2|x)$
  - Otherwise decide ω_2
- Probability of error:

$$P(error|x) = \min[P(\omega_1|x), P(\omega_2|x)]$$



Generalization of the preceding ideas

- Use of more than one feature
  - **Feature vector** $\mathbf{x}$ lies in a d-dimentional **feature space**
- Use more than two states of nature
- Allow actions and not only decide on the state of nature
- Introduce a loss function which is more general than the probability of error

Loss function

- Allowing actions other than classification allows the possibility of rejection
  - Refuse to make a decision in close or bad cases
- **Loss function** states how costly each action is
  - Let $\omega_1, ..., \omega_c$ be the set of c categories (classes, states of nature)
  - Let $\alpha_1, ..., \alpha_a$ be the set of a possible actions
  - $\lambda(\alpha_i|\omega_j)$ describes the loss incurred for taking action α_i when the category is ω_j

Bayes risk

- **Decision rule** is a function $\alpha(\mathbf{x})$ assuming one of the α values $\alpha_1, ..., \alpha_a$

- **Overall risk** R is the expected loss associated with a given decision rule:

$$R = \int R(\alpha(\mathbf{x})|\mathbf{x})p(\mathbf{x})d\mathbf{x},$$

where the **conditional risk:**

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

- **Bayes decision rule:** To minimize R, select the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum

Two-category classification

- Assumptions:
  - α_1: deciding ω_1
  - α_2: deciding ω_2
  - $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$: loss incurred for deciding ω_i when the true category is ω_j
- Conditional risk:

$$R(\alpha_1|\mathbf{x}) = \lambda_{11}P(\omega_1|\mathbf{x}) + \lambda_{12}P(\omega_2|\mathbf{x})$$

$$R(\alpha_2|\mathbf{x}) = \lambda_{21}P(\omega_1|\mathbf{x}) + \lambda_{22}P(\omega_2|\mathbf{x})$$

$$\Downarrow$$

- **Likelihood ratio**-based decision rule (assuming $\lambda_{21} > \lambda_{11}$):
  Decide ω_1 if

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

Minimum error-rate classification

- Actions are often decisions on classes
- If action α_i is taken and the true category is ω_j, then:
  - The decision is correct if $i = j$, and in error if $i \neq j$
- **Minimum error-rate classification:** decision rule must minimize the probability of error, i.e., the **error rate**
- **Zero-one** loss function:

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0, & i = j, \\ 1, & i \neq j \end{cases}$$

- Decision rule in this case:
  - Decide ω_i if $P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x})$ for all $j \neq i$

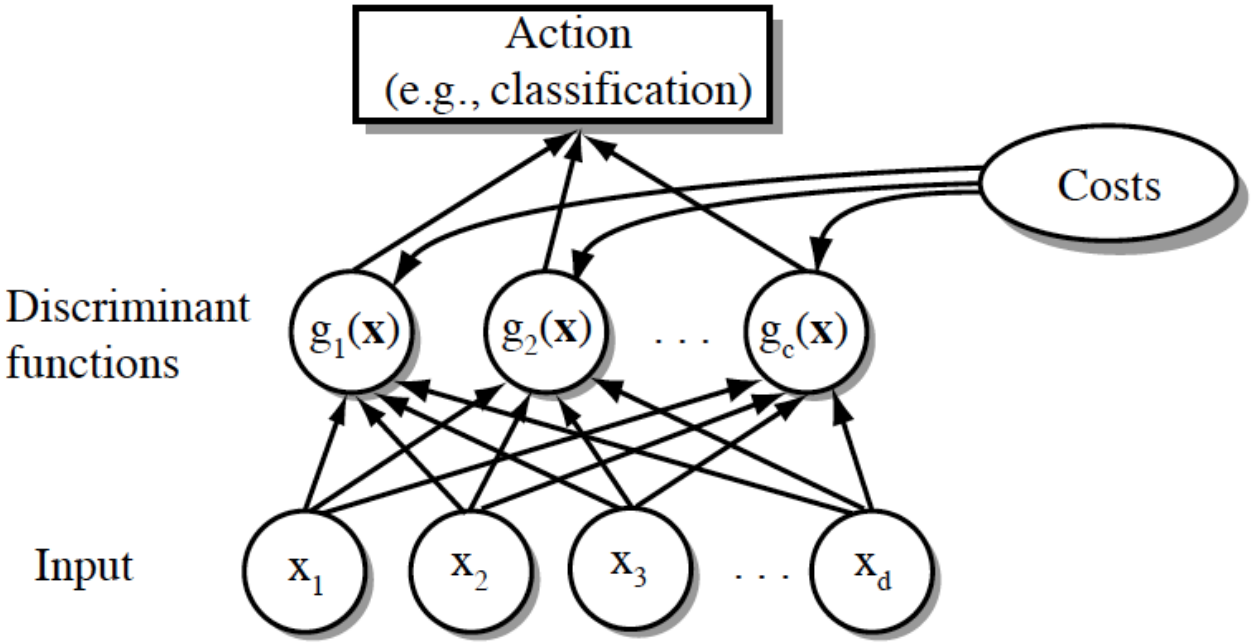
Discriminant versus generative classification models

- **Fundamental difference** between discriminative and generative models:
  - **Discriminative models** learn the **boundary** between classes
  - **Generative models** model the **distribution** of individual classes

Discriminant model

- Set of **discriminant functions** $g_i(\mathbf{x}), i = 1, ..., c$
- The classifier assigns a feature vector $\mathbf{x}$ to class ω_i if:

$$g_i(\mathbf{x}) > g_j(\mathbf{x}), \quad \text{for all } j \neq i$$



Discriminant functions

- A Bayes classifier is easily represented in terms of discriminant functions
- For the general case with risks, we can let:

$$g_i(\mathbf{x}) = -R(\alpha_i|\mathbf{x})$$

- For the minimum-error-rate case, we can use:

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

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

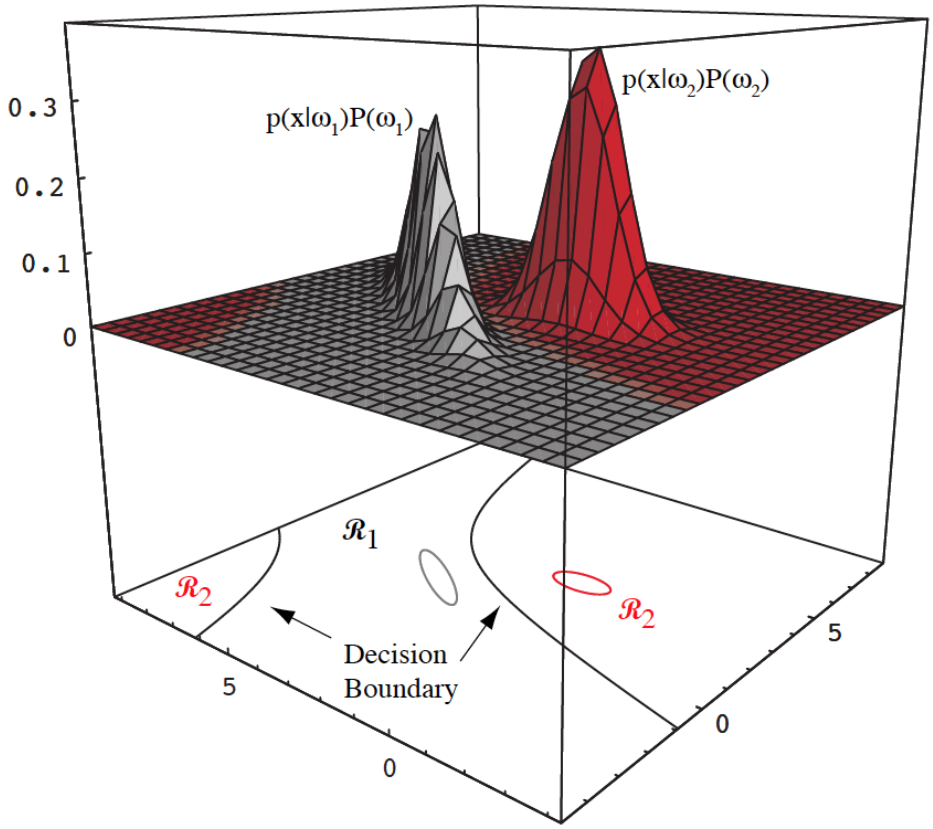
$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i),$$

where ln denotes natural logarithm

Decision regions & decision boundaries

- The choice of discriminant functions is not unique
- However, the decision rules are equivalent
- The effect of any decision rule is to divide the feature space into c **decision regions**, $\mathcal{R}_1, ..., \mathcal{R}_c$
- If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is in $\mathcal{R}_i$
  - Means assign $\mathbf{x}$ to ω_i
- The regions are separated by **decision boundaries**

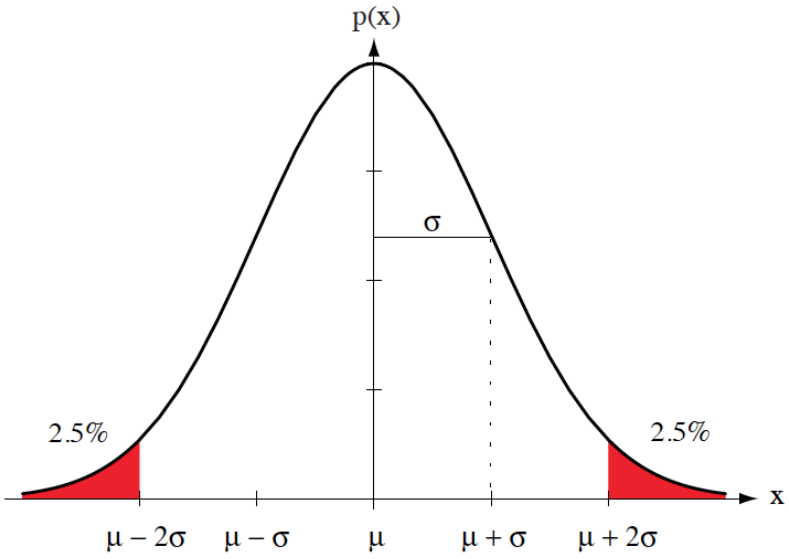
Decision regions & decision boundaries



- Two-dimensional two-category classifier
- Decision boundary consists of two hyperbolas

The normal density

- Structure of a Bayes classifier is determined by the conditional densities $p(\mathbf{x}|\omega_i)$, as well as by the prior probabilities
- The most popular density function: normal, or Gaussian density
  - Analytical tractability
  - A lot of processes are asymptotically Gaussian
    - But not all!



The normal density

- Univariate normal density:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

where the mean or expected value of x

$$\mu = \int_{-\infty}^{\infty} x p(x) dx$$

and the variance of x:

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx$$

- The univariate normal density $\mathcal{N}(\mu, \sigma^2)$ is completely specified by two parameters: μ and σ^2

Multivariate normal density

- Multivariate normal density:

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

  where:

  - $\mathbf{x}$ is a d-component feature vector
  - $\boldsymbol{\mu}$ is a d-component **mean** vector
  - Σ is the d-by-d **covariance matrix**
  - $|\Sigma|$ and Σ^{-1} are its determinant and inverse, respectively

- The multivariate normal density is completely specified by $d + d(d+1)/2$ parameters: $\boldsymbol{\mu}$ and the independent elements of Σ

Maximum likelihood and Bayesian parameter estimation

- Bayes classifier: if we know class-conditional densities and prior probabilities, we could design an optimal classifier
- In practice, no info about the complete probabilistic structure
  - But general knowledge about the situation &
  - **Training samples** - particular representatives for each category
- Design a classifier from training samples
  - No problems with prior estimation
  - Training data are often too small for class-conditional estimation
    - Especially if the dimensionality of the feature space is large

Maximum likelihood and Bayesian parameter estimation

- A priori information about the problem
  - Example: we can assume that $p(\mathbf{x}|\omega_i)$ is a normal density
  - The problem is reduced to estimating the parameters $\boldsymbol{\mu}$ and Σ
- Parameter estimation techniques:
  - Maximum likelihood
  - Bayesian estimation
- Results are (in some cases) nearly identical, but the approaches are different

Parameter estimation

- **Maximum likelihood:**
  - Values of parameters are fixed but unknown
  - Best estimates are obtained by maximizing the probability of obtaining the samples observed
- **Bayesian estimation:**
  - Parameters as random variables having some known a priori distribution
  - Observation of the samples converts this to a posterior density

Maximum likelihood estimation

- Advantages:
  - Has good convergence properties as the number of training samples increases
  - Simpler than alternative methods

Maximum likelihood estimation - General principle

- Assume we have c classes and

$$p(\mathbf{x}|\omega_j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$$

$$p(\mathbf{x}|\omega_j) \equiv p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j),$$

  where $\boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \Sigma_j)$

- Use the information provided by the training samples to estimate $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$ associated with each category

- c separate problems (one problem per class)

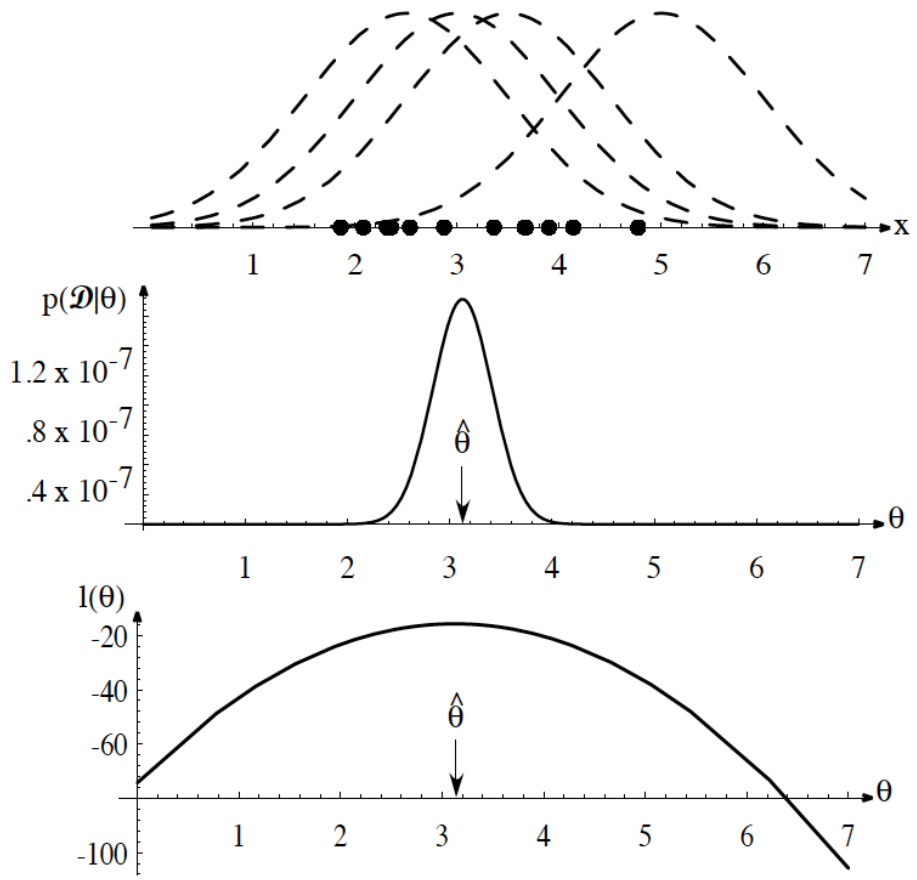
Maximum likelihood estimation - General principle

- Use a set $\mathcal{D}$ of n training samples $x_1, x_2, ..., x_n$ drawn independently from $p(\mathbf{x}|\boldsymbol{\theta})$ to estimate the unknown $\boldsymbol{\theta}$:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta})$$

- $p(\mathcal{D}|\boldsymbol{\theta})$ is called the **likelihood** of $\boldsymbol{\theta}$ w.r.t. the set of samples
- **Maximum likelihood** estimate of $\boldsymbol{\theta}$ is the value $\hat{\boldsymbol{\theta}}$ that maximizes $p(\mathcal{D}|\boldsymbol{\theta})$
  - This value best agrees with the training samples

Maximum likelihood estimation - General principle



- Top: Training points, assumed to be drawn from a Gaussian of a particular variance with unknown mean
- Middle: likelihood as a function of mean
- Bottom: Log-likelihood function

$$l(\boldsymbol{\theta}) \equiv \ln p(\mathcal{D}|\boldsymbol{\theta})$$

Maximum likelihood estimation - General principle

- Let $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)^T$ be a p-component vector
- The gradient:

$$\nabla_{\boldsymbol{\theta}} \equiv \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix}$$

- Problem statement: determine $\boldsymbol{\theta}$ that maximizes the log-likelihood:

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

Set of necessary conditions for an optimum:

$$\nabla_{\boldsymbol{\theta}} l = \sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k | \boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} l = 0$$

Maximum likelihood estimation - Example

- The samples are drawn from a normal distribution with unknown $\boldsymbol{\mu}$ and known Σ

$$\ln p(\mathbf{x}_k|\boldsymbol{\mu}) = -\frac{1}{2}\ln[(2\pi)^d|\Sigma|] - \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}_k - \boldsymbol{\mu})$$

$$\Downarrow$$

$$\nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k|\boldsymbol{\mu}) = \Sigma^{-1}(\mathbf{x}_k - \boldsymbol{\mu})$$

- The maximum likelihood estimate for $\boldsymbol{\mu}$ must satisfy:

$$\sum_{k=1}^{n} \Sigma^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}) = 0$$

$$\Downarrow$$

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k \quad \text{(just sample mean)}$$

Maximum likelihood estimation - Example

- If we consider a multivariate normal case with unknown mean and covariance:
  - The maximum likelihood estimate for $\boldsymbol{\mu}$ is the sample mean = arithmetic average of the training samples
  - The maximum likelihood estimate for the covariance matrix Σ is a sample covariance matrix:

$$\mathbf{C} = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T$$

Maximum likelihood - Example with real world data

- Hyperspectral image classification
  - Image acquired by the ROSIS-03 optical sensor over the University of Pavia, Italy
  - Spatial dimension: 610 $\times$ 340 pixels
  - Spatial resolution: 1.3m per pixel
  - Spectral dimension: 103 spectral channels (0.43-0.86 μm)

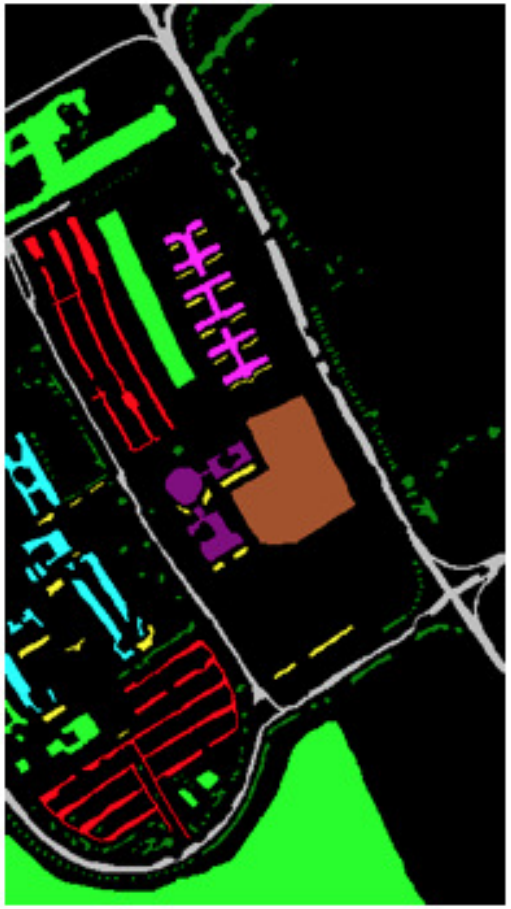
Maximum likelihood - Hyperspectral image classification

Input image (103 spectral channels)

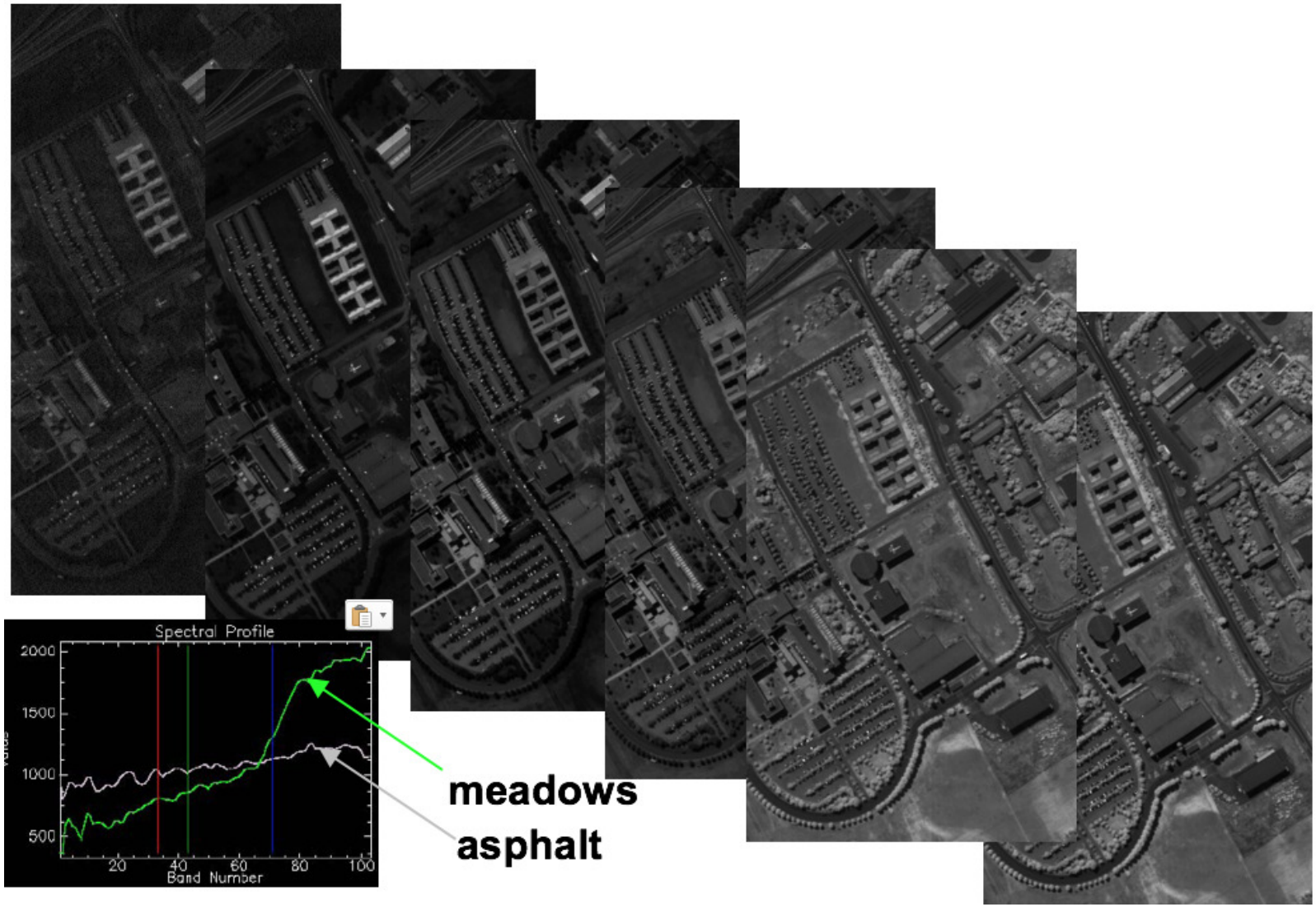
Task: assign every pixel to one of the nine classes

alphalt
meadows
gravel
trees
metal sheets
bare soil
bitumen
bricks
shadows

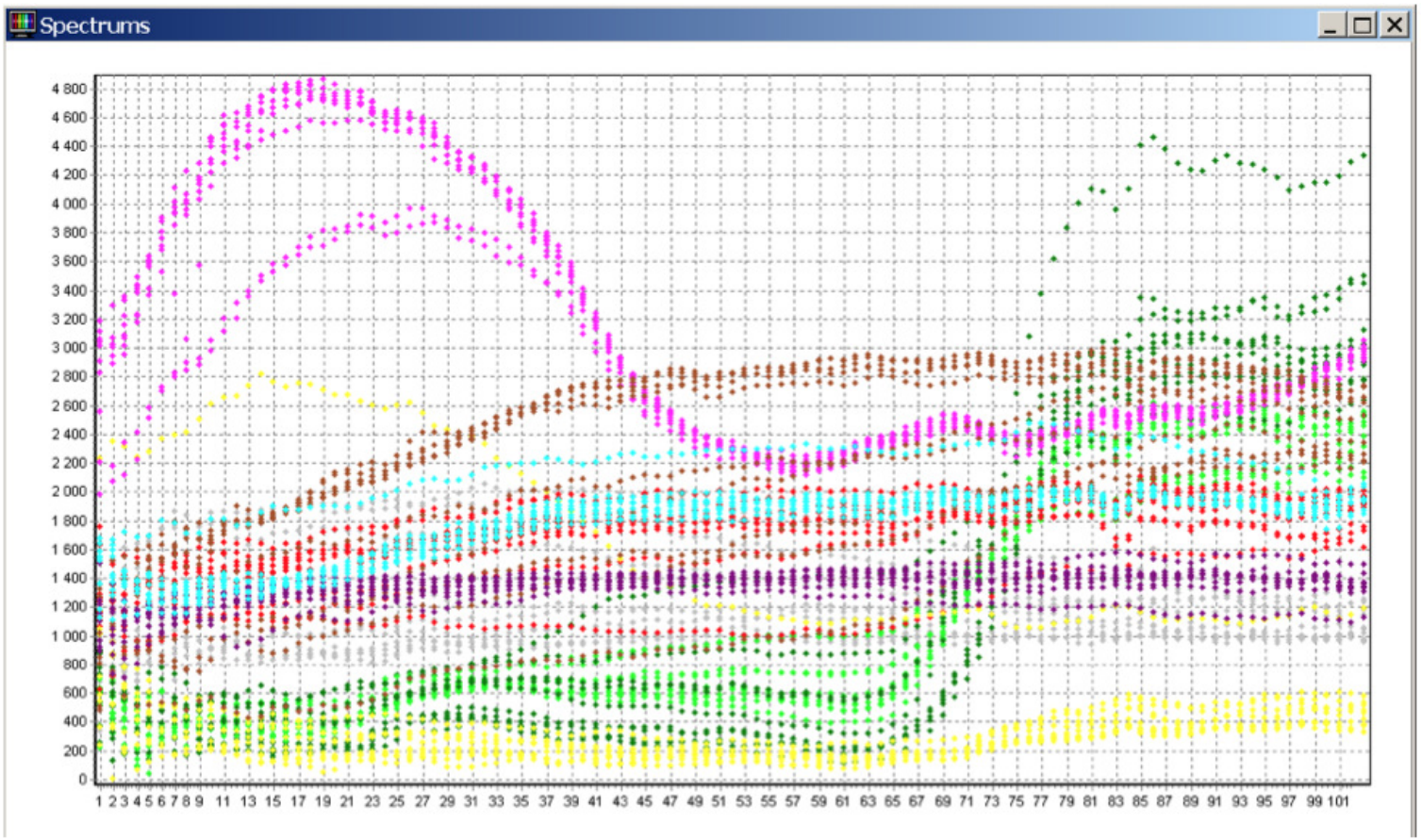
Reference data



Spectral context



Spectral context

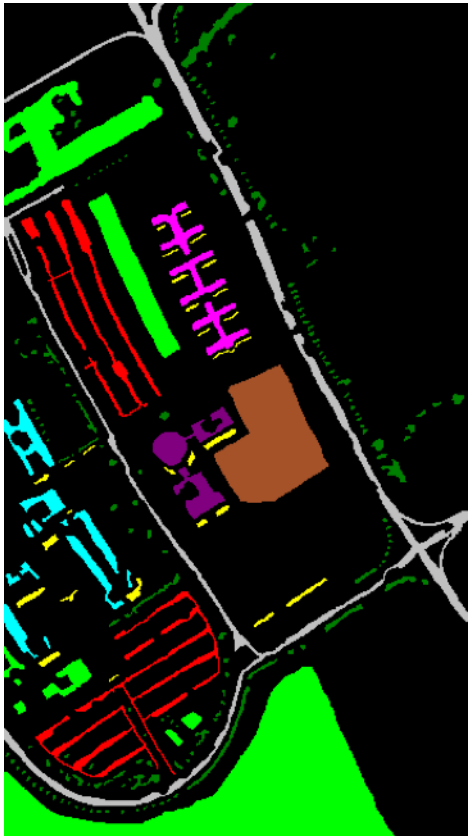


alphalt, meadows, gravel, trees, metal sheets, bare soil, bitumen, bricks, shadows

Maximum likelihood classification

- **Feature vector**: a vector of radiance values for each pixel
  - 103 spectral bands $\Rightarrow$ dimensionality of the vector $d = 103$
- Samples of each class k are assumed to have a Gaussian distribution
- Estimate parameters of distributions from the training samples, using the maximum likelihood estimates

Maximum likelihood classification



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

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

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: $p << m_k$
  - m_k: number of training samples for class k
- Dimensionality reduction must be performed to reduce d
  - The first 3 bands on the 103-band image are omitted
  - Averaging over every 10 bands $\Rightarrow$ new $d = 10$

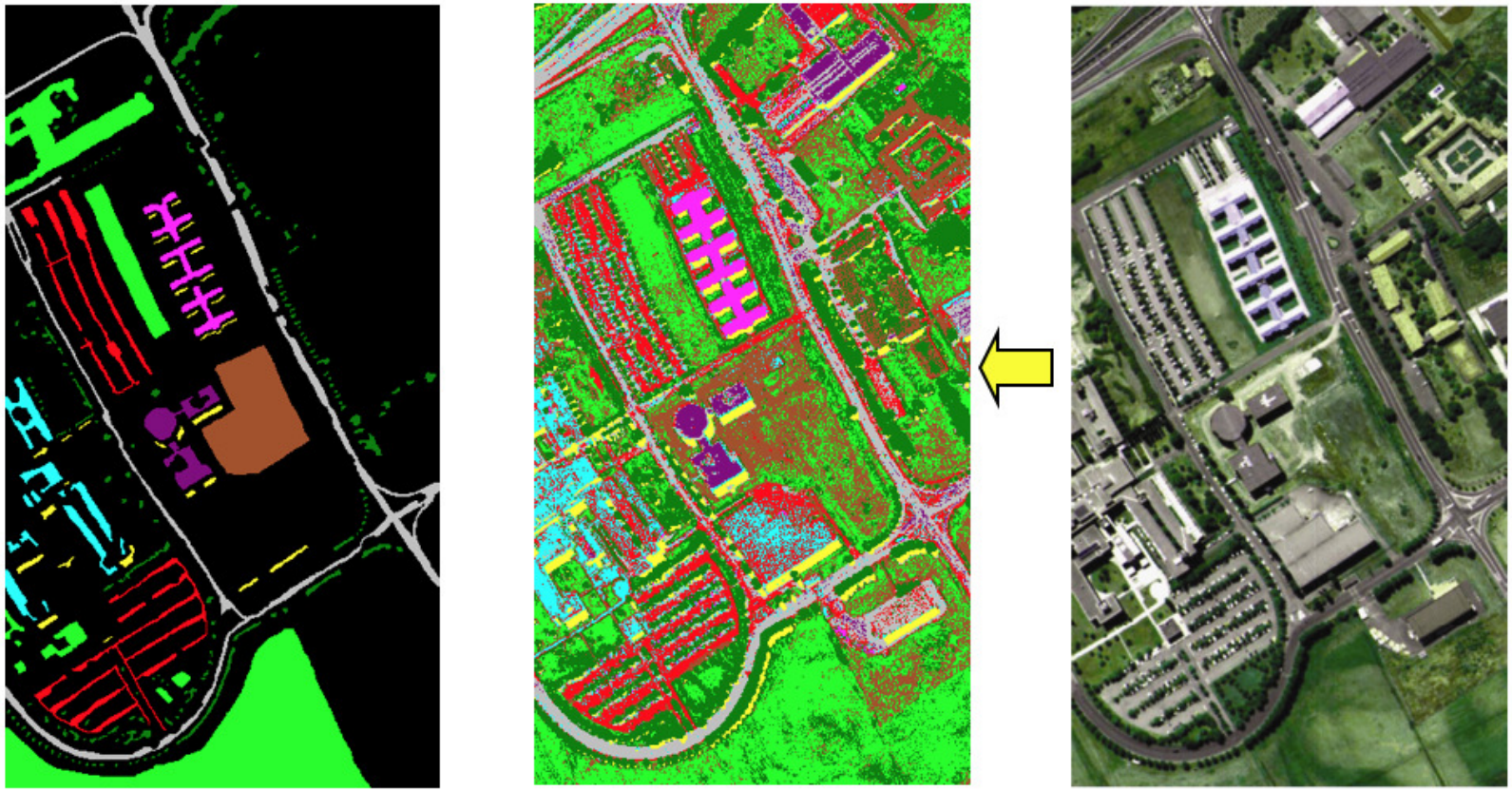
Maximum likelihood classification

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

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

$$P(\omega_k) = \frac{m_k}{m}, \quad m \text{ - total number of training samples}$$

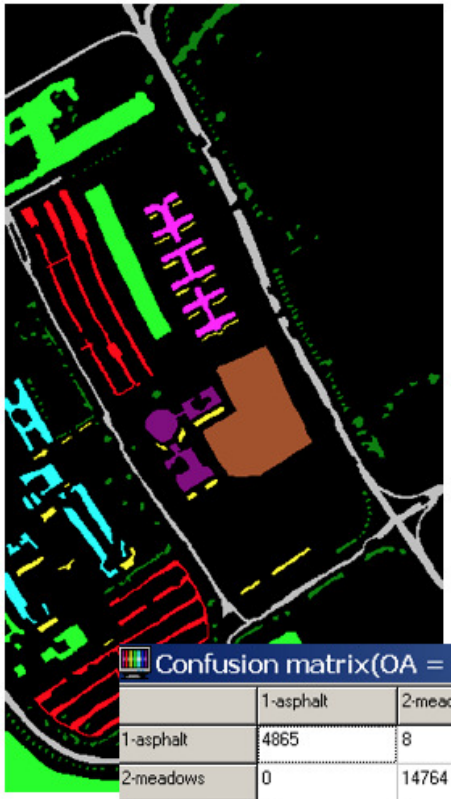
Maximum likelihood classification



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
- More advanced feature extraction (dimensionality reduction) method can be used

  $\Rightarrow$ Accuracies can be further improved

- Computational complexity:
  - For learning: $\mathcal{O}(d^2 m)$
  - For classification of 1 sample: $\mathcal{O}(cd^2)$
  - Parallel implementations significantly reduce processing time

Bayesian parameter estimation

- The main difference w.r.t. maximum likelihood (ML) is that in the Bayesian case $\boldsymbol{\theta}$ **is a random variable**
- Goal: estimate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$, using Bayesian theory:

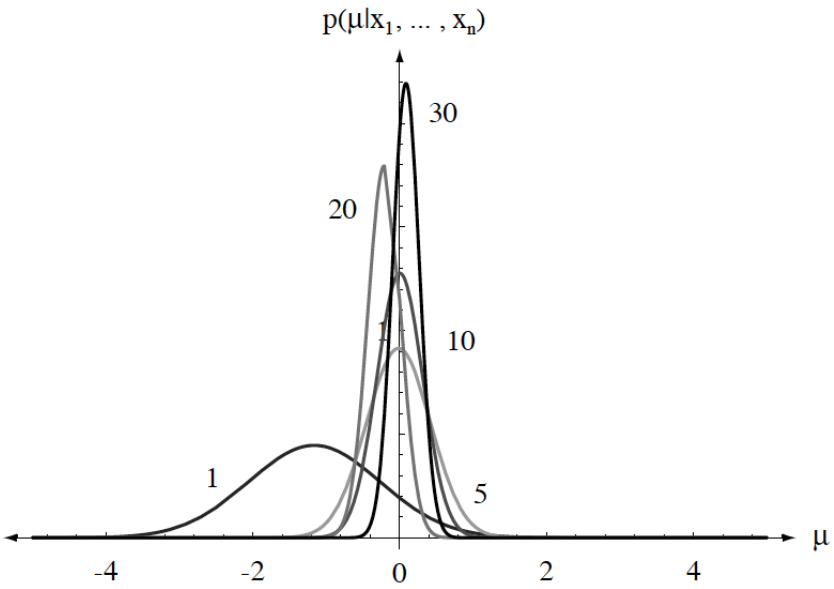
$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$$

- ML produces the best estimate (a number), while Bayes produces the distribution of θ
  - In addition to the most probable estimate, we obtain a characterization of the **uncertainty**

Bayesian parameter estimation - Example

- Gaussian univariate case where μ is the only unknown parameter
- Bayes formula gives:

$$p(\mu|\mathcal{D}) = \frac{p(\mathcal{D}|\mu)p(\mu)}{\int p(\mathcal{D}|\mu)p(\mu)d\mu}$$



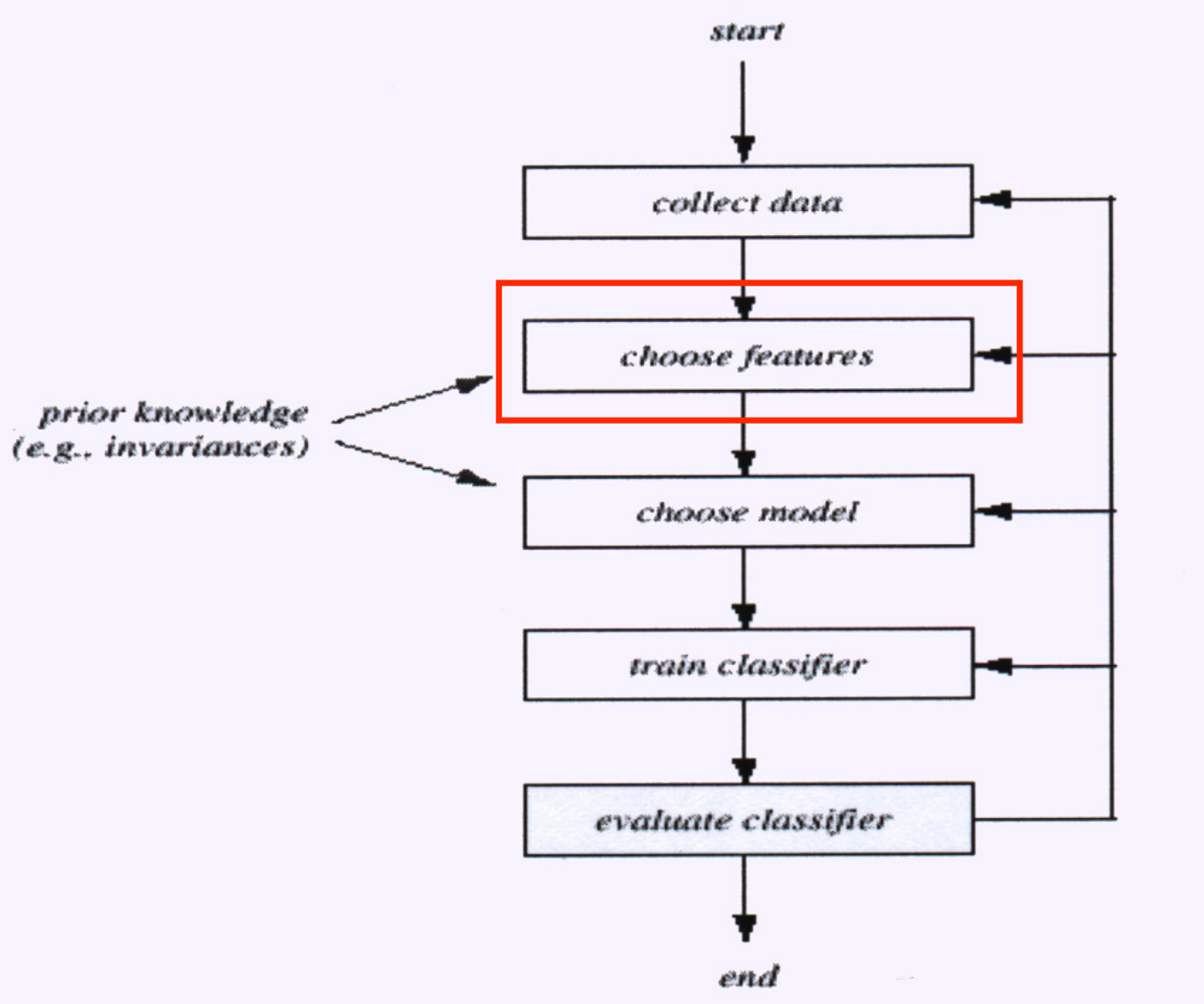
- The posterior distribution estimates are labelled by the number of training samples used in the estimation

Concluding remarks

- There are strong theoretical arguments supporting Bayesian estimation
- In practice Maximum likelihood is one of the simplest techniques for parameter estimation
  - Often leads to classifiers nearly as accurate as with the Bayesian estimation

! Crucial factor to boost performance of machine learning methods: choice of the **feature space**

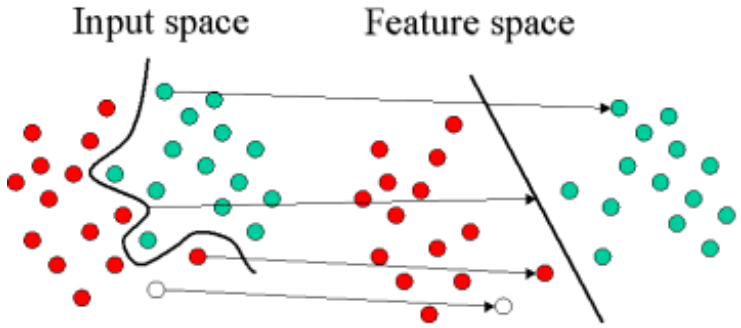
Why do we need feature extraction?



Motivation

- **Main motivation:** get out most of the data

- For classification task: find a space where samples from different classes are well separable

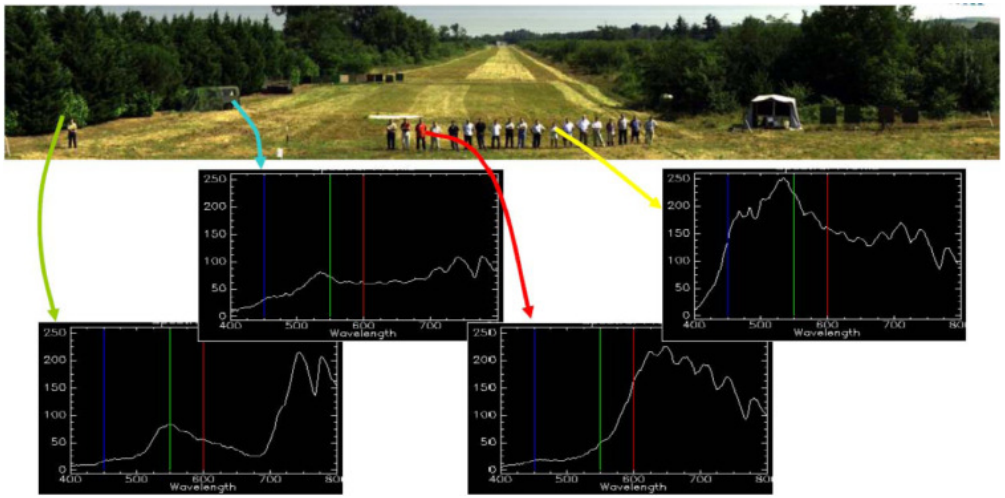


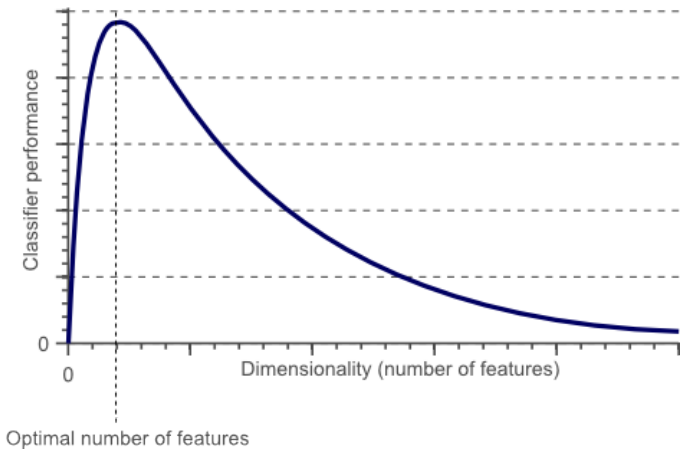
Objectives:

- Reduce computational load of the classifier
- Increase data consistency
- Incorporate different sources of information into a feature vector: spectral, spatial, multisource, ...

Motivation - Curse of dimensionality

- Too few features do not allow to discriminate between classes
  - In the color image, both trees and a truck are green
- As the dimensionality of the feature space increases, the classifier's performance increases until the optimal number of features is reached
- Further increasing the dimensionality without increasing the number of training samples yields a performance decrease





How to reduce data dimensions?

Principal component analysis

Convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables, called **principal components**

Discriminant analysis

Find the best set of vectors which best separates the patterns

Principal component analysis

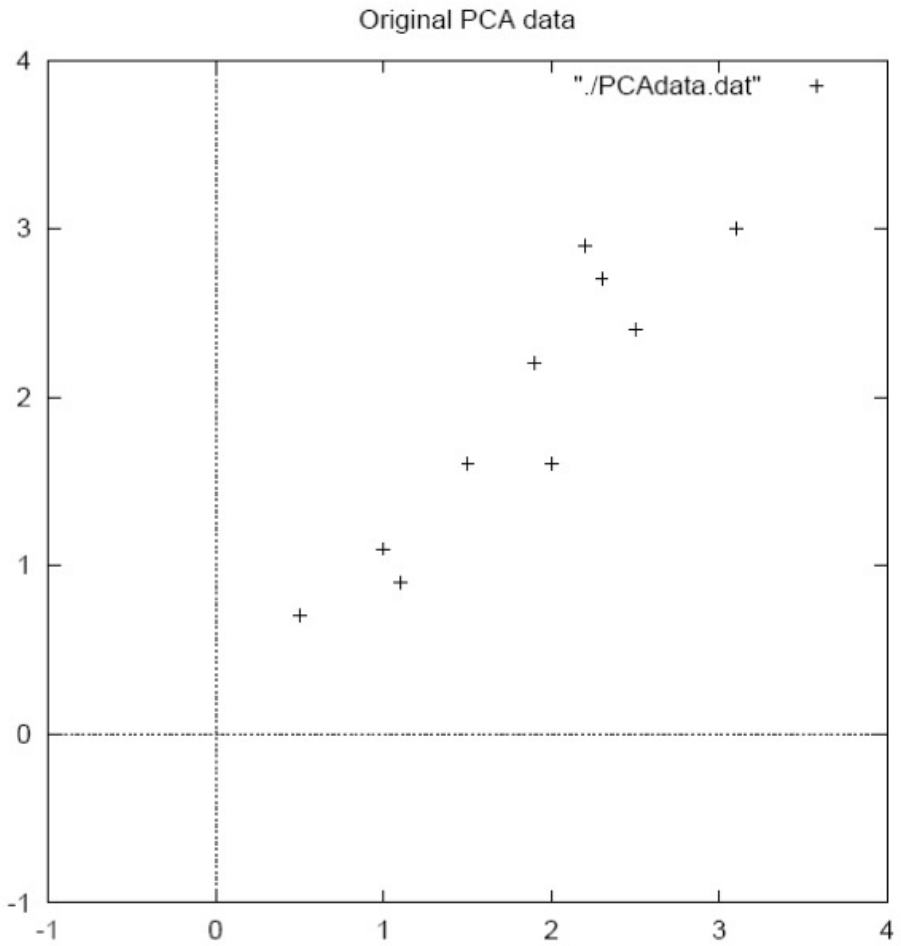
- **Goal**: represent data in a space that best describes the variation in a sum-squared error sense
- Projection onto eigenvectors that correspond to the first few largest eigenvalues of the covariance matrix
  - d-dimensional data are represented in a lower-dimensional space
  - Reduces the space and time complexities
- Intuitive introduction: `http://www.youtube.com/watch?v=BfTMmoDFXyE&feature=related`

Principal component analysis

- **Step 1**: Get some data

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9



Principal component analysis

- **Step 2**: Subtract the mean
  - From each of the data dimensions (from x- and y-dimension)

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Principal component analysis

- **Step 3**: Calculate the covariance matrix

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

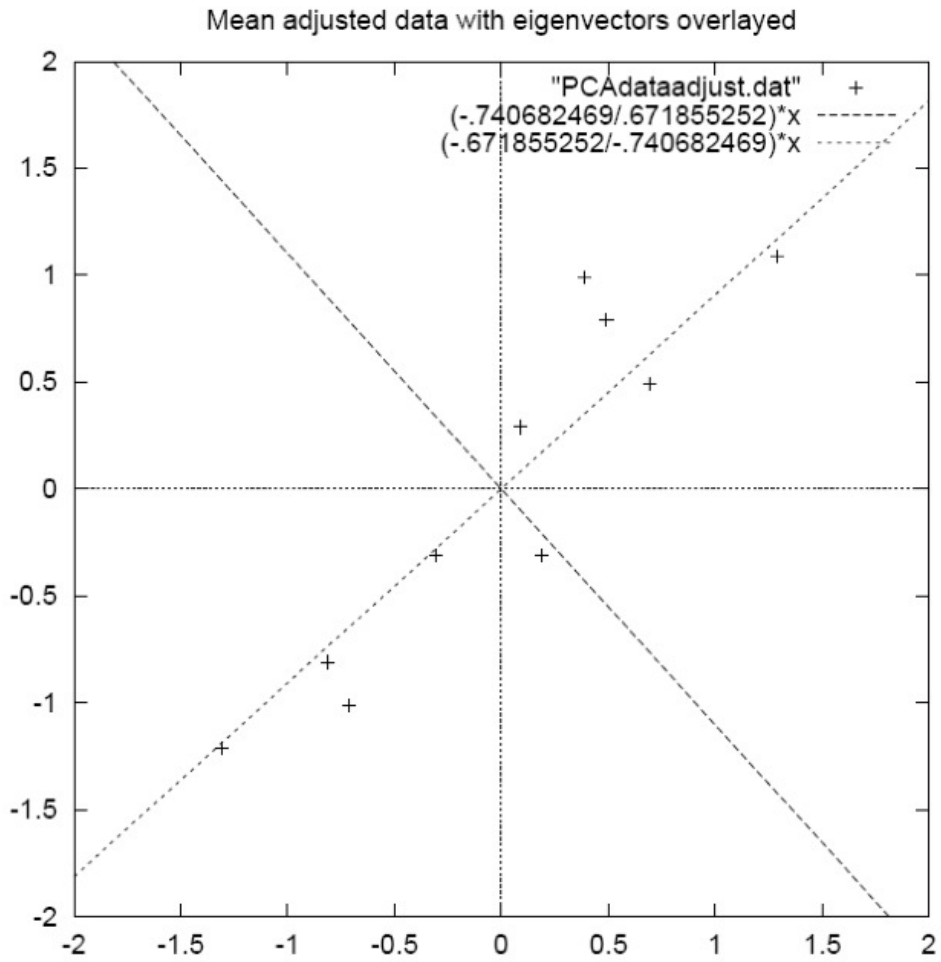
Principal component analysis

- **Step 4**: Calculate the unit eigenvectors and eigenvalues of the covariance matrix

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$



Principal component analysis

- The 1st eigenvector (principle component) shows how data in two dimensions are related along the eigenvector line
- The 2nd eigenvector shows that all the points are off to the side of the main line by some amount

- Eigenvectors are lines that characterize the data
- The next steps: transforming the data so that it is expressed in terms of these lines

Principal component analysis

- **Step 5**: Choose components and form a feature vector
  - Order eigenvectors by eigenvalues
    - This gives the components in order of significance
    - You can decide to ignore the components of lesser significance $\Rightarrow$ final data will have less dimensions ($p < d$)
  - Form a projection vector (matrix of vectors):

$$FeatureVector = (eig_1 \;\; eig_2 \;\; eig_3)$$

- For our example, two projection vectors are possible:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Principal component analysis

- **Step 6**: Derive the new dataset:

$$FinalData = ProjectionVector^T \times RowDataAdjust$$

where *RowDataAdjust* is the mean-adjusted data transposed

- It will give us the original data solely in terms of the vectors we chose

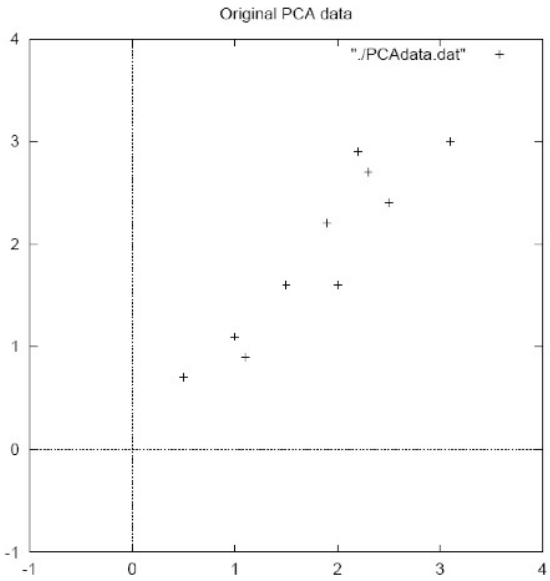
Principal component analysis

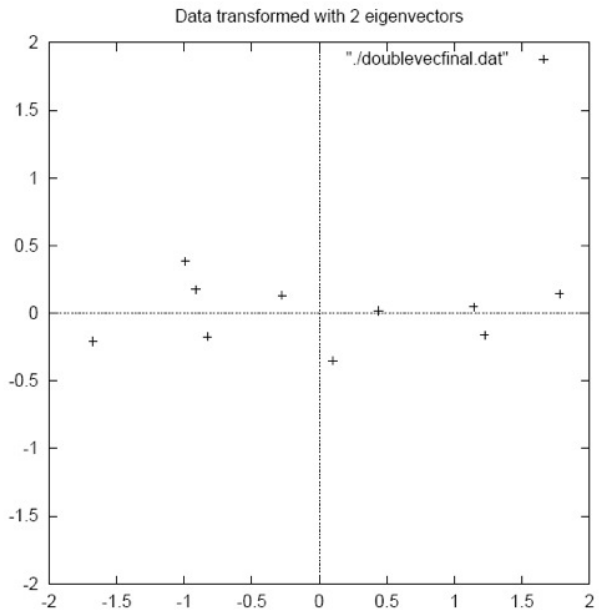
Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287



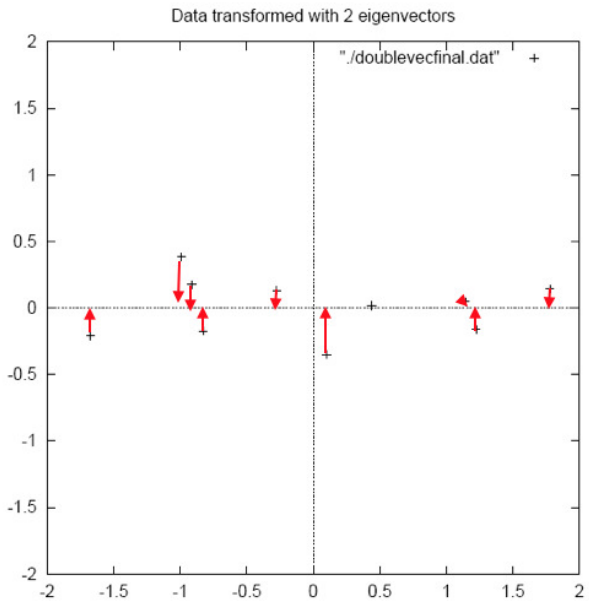


Principal component analysis (PCA)

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

- If only one eigenvector was kept, the transformed data will have only one dimension



Example of PCA for hyperspectral image analysis

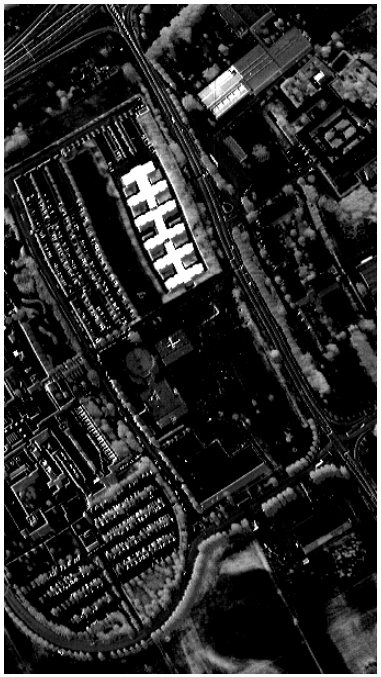
- Principal component analysis in the spectral space
  - Principal components (PCs) 1-3 contain 97% of information from original 103 channels

Color image

PC1

PC2

PC3



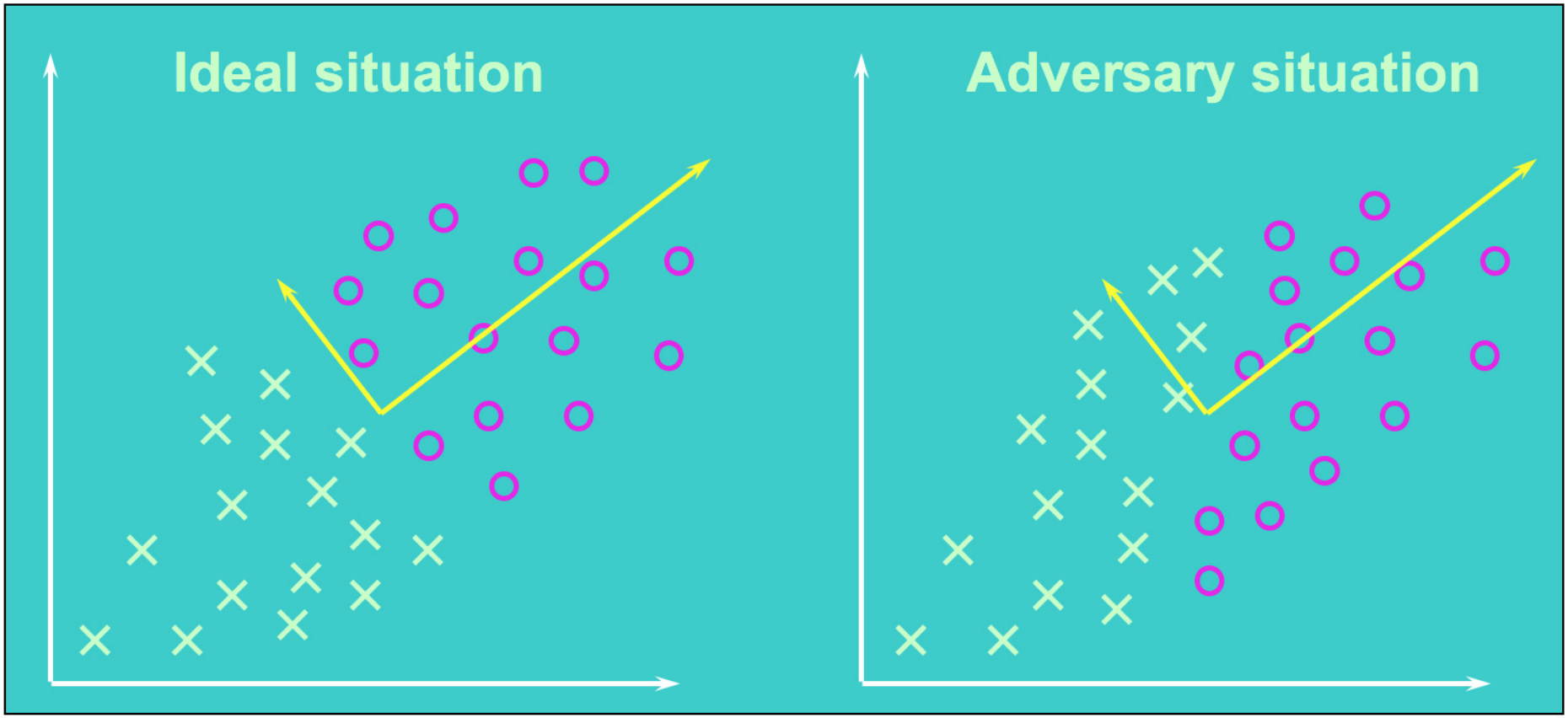
PC4

Principal component analysis

Exercise: PCA

Principal component analysis

- Projection onto eigenvectors that correspond to the first few largest eigenvalues of the covariance matrix

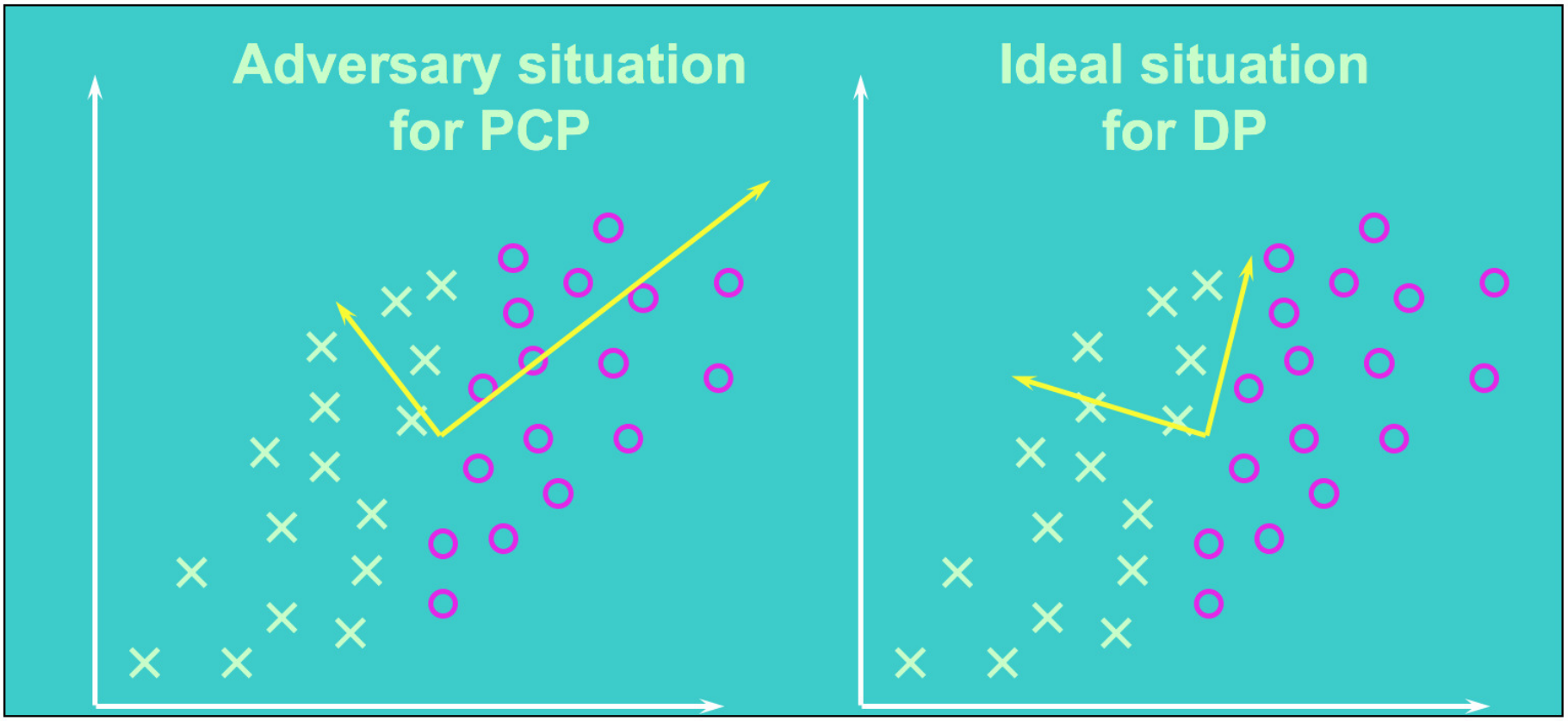


Discriminant analysis

- PCA seeks directions that are efficient for **representation**
  - *Unsupervised technique*
- Discriminant analysis seeks directions that are efficient for **discrimination**
  - *Supervised technique*

Discriminant analysis

- Projection onto directions that can best separate data of different classes



Discriminant analysis

- Theory of Fisher linear discriminant: `http://www.csd.uwo.ca/~olga/Courses//CS434a_541a//Lecture8.pdf`
- Project on line in the direction v which maximizes:

want projected means are far from each other

$$J(v) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

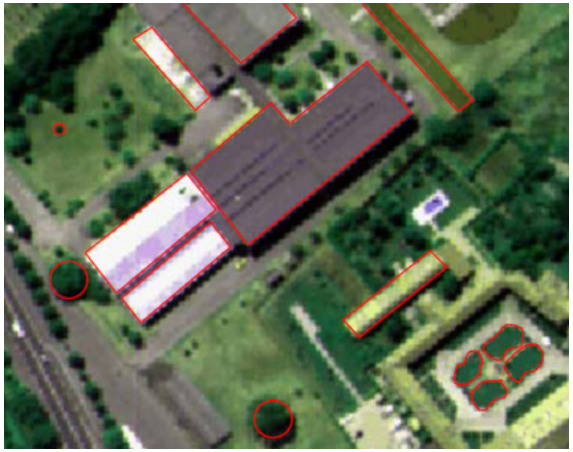
want scatter in class 1 is as small as possible, i.e. samples of class 1 cluster around the projected mean $\tilde{\mu}_1$

want scatter in class 2 is as small as possible, i.e. samples of class 2 cluster around the projected mean $\tilde{\mu}_2$

- Main drawback: in most real-life cases, projection to even the best line results in unseparable projected samples

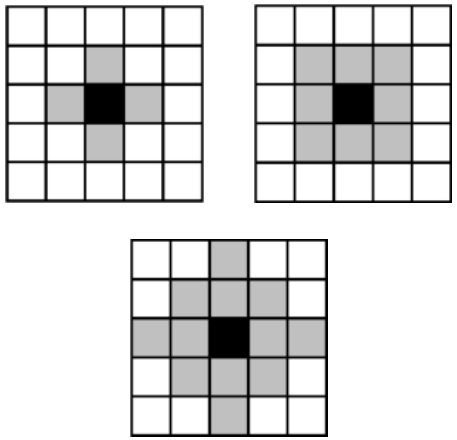
How to include spatial information for image classification?

- By simply looking at a grey pixel, we cannot say if it belongs to a *building* or a *road*
- We guess a category by considering spatial/contextual information
- How can a classifier consider this rich source of information?



Approaches to extract spatial info

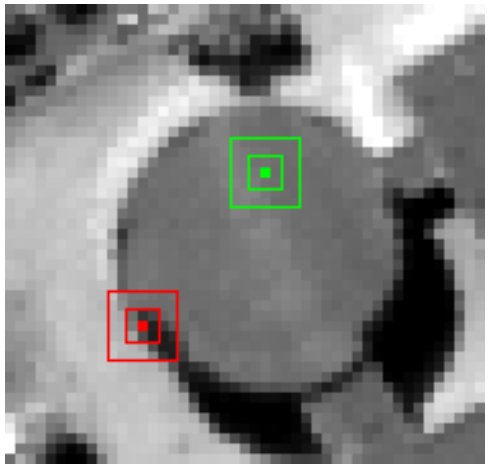
1. Closest fixed neighborhoods
   - Markov Random Field [Pony00, Jackson02, Farag05]
   - Contextual features [Camps-Valls06]
     - Spectral content +
     - Spatial content (e.g. mean or standard deviation per spectral band)



+ Simplicity
− Imprecision at the border of regions

Approaches to extract spatial info

1. Closest fixed neighborhoods
   - Markov Random Field [Pony00, Jackson02, Farag05]
   - Contextual features [Camps-Valls06]
     - Spectral content +
     - Spatial content (e.g. mean or standard deviation per spectral band)



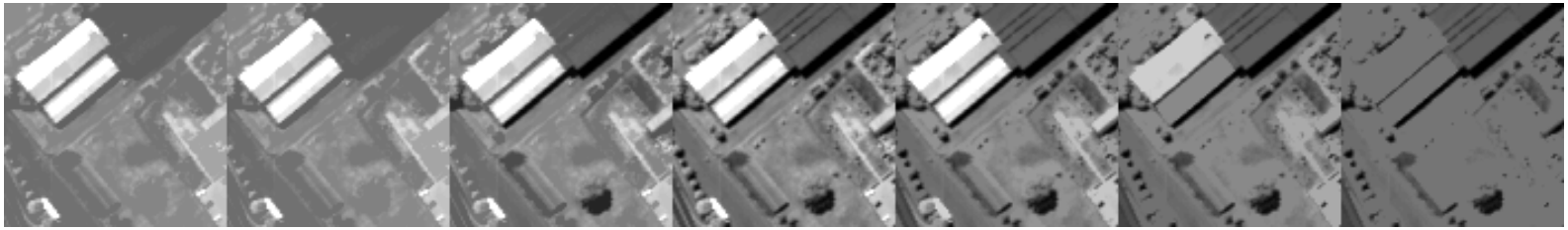
\+ Simplicity
− Imprecision at the border of regions

Approaches to extract spatial info

2. Morphological and area filtering

- Morphological profiles [Pesaresi01, Dell'Acqua04, Benediktsson05]
- Self-complementary area filtering [Fauvel07]
- Attribute profiles [Ghamisi15, Cavallaro17]

+ Neighborhoods are adapted to the structures
+ Non-linear operators ⇒ do not blur the edges as convolutions do
− Neighborhoods are scale dependent



← Closing – Original – Opening →

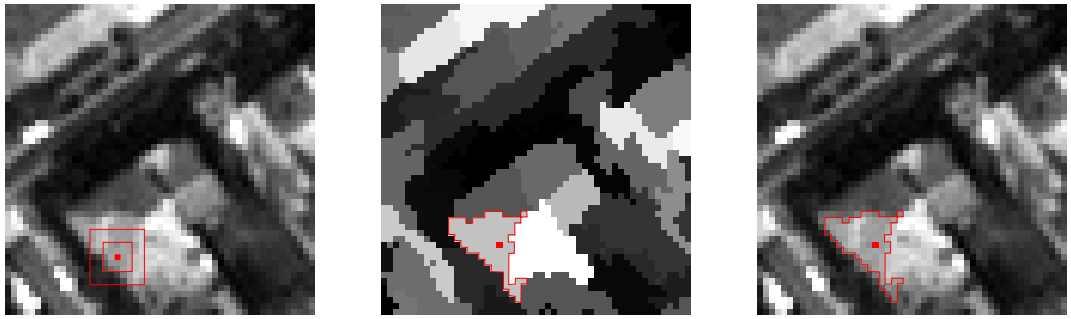
Course on mathematical morphology:
`http://www-sop.inria.fr/members/Yuliya.Tarabalka/teaching.htm`

Approaches to extract spatial info

2. Morphological and area filtering

- Morphological profiles [Pesaresi01, Dell'Acqua04, Benediktsson05]
- Self-complementary area filtering [Fauvel07]
- Attribute profiles [Ghamisi15, Cavallaro17]

+ Neighborhoods are adapted to the structures
+ Non-linear operators ⇒ do not blur the edges as convolutions do
− Neighborhoods are scale dependent



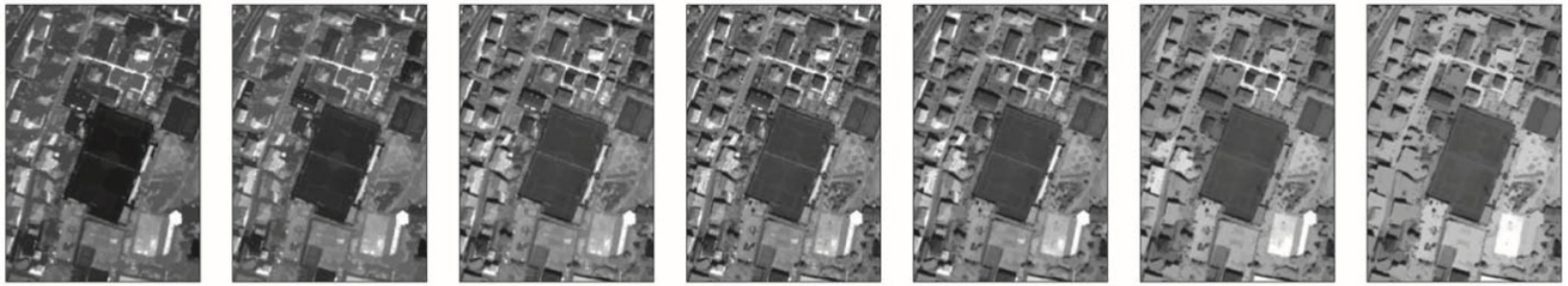
Course on mathematical morphology:
`http://www-sop.inria.fr/members/Yuliya.Tarabalka/teaching.htm`

Approaches to extract spatial info

2. Morphological and area filtering

- Morphological profiles [Pesaresi01, Dell'Acqua04, Benediktsson05]
- Self-complementary area filtering [Fauvel07]
- Attribute profiles [Ghamisi15, Cavallaro17]

+ Neighborhoods are adapted to the structures
+ Non-linear operators ⇒ do not blur the edges as convolutions do
− Neighborhoods are scale dependent

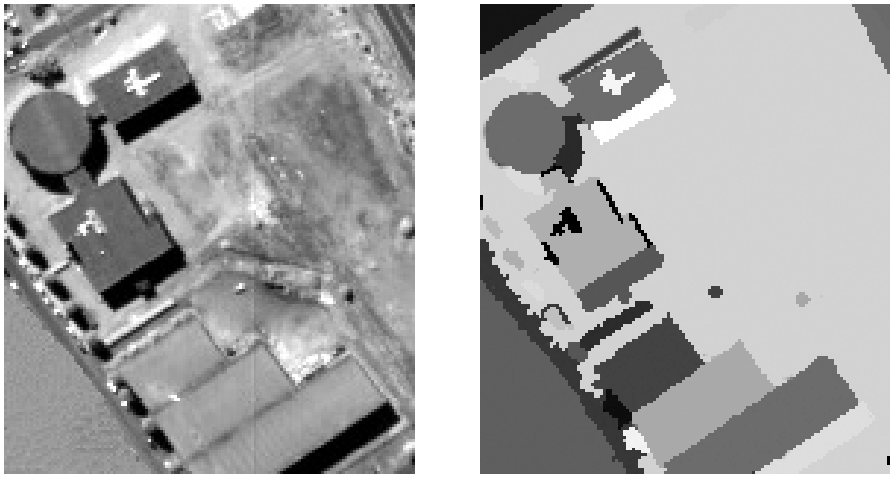


Course on mathematical morphology:
`http://www-sop.inria.fr/members/Yuliya.Tarabalka/teaching.htm`

Approaches to extract spatial info

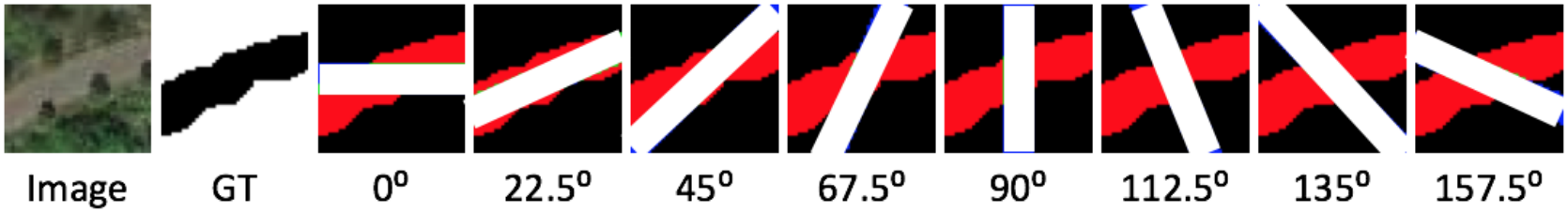
3. Superpixels derived from segmentation
   - Extraction and Classification of Homogeneous Objects [Kettig76]
   - ...
   - Multiscale segmentation, then features are derived from the regions [Linden07, Huang09]

     \+ Flexible
     − Computationally demanding
     − Difficult to scale/parallelize



Approaches to extract spatial info

4. Features handcrafted for a particular application
   - Example 1: Line templates with different orientations for road detection [Jeong15]
   - Example 2: Rectangular templates for building detection [Garcin01]
     + Can model complex shape
     – Lack of genericity
     – Computationally demanding



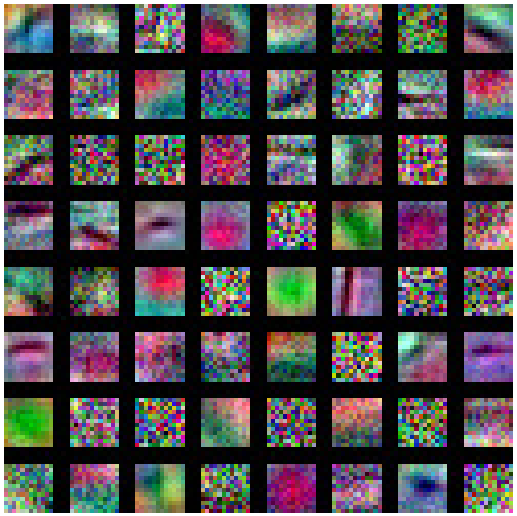
Approaches to extract spatial info

4. Features handcrafted for a particular application
   - Example 1: Line templates with different orientations for road detection [Jeong15]
   - Example 2: Rectangular templates for building detection [Garcin01]
     - \+ Can model complex shape
     - – Lack of genericity
     - – Computationally demanding

Modern trend & Conclusions

Deep learning:

- Automatically learn features if a lot of training data are available

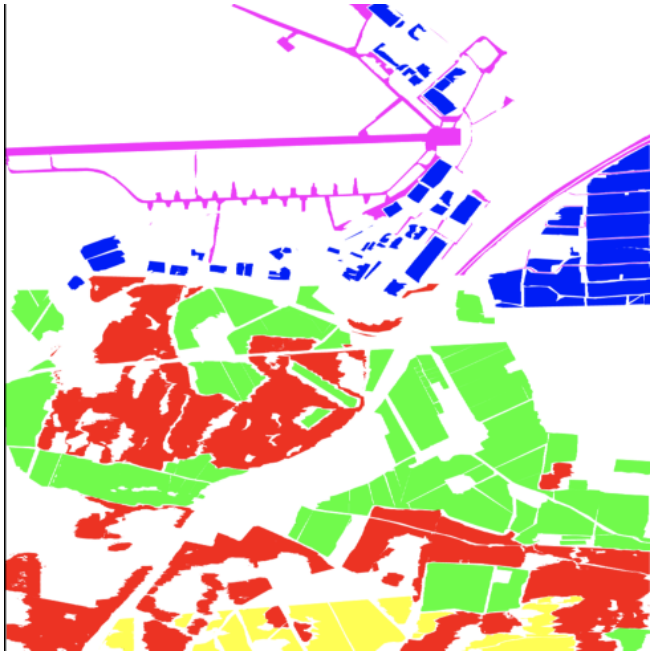


Advice:

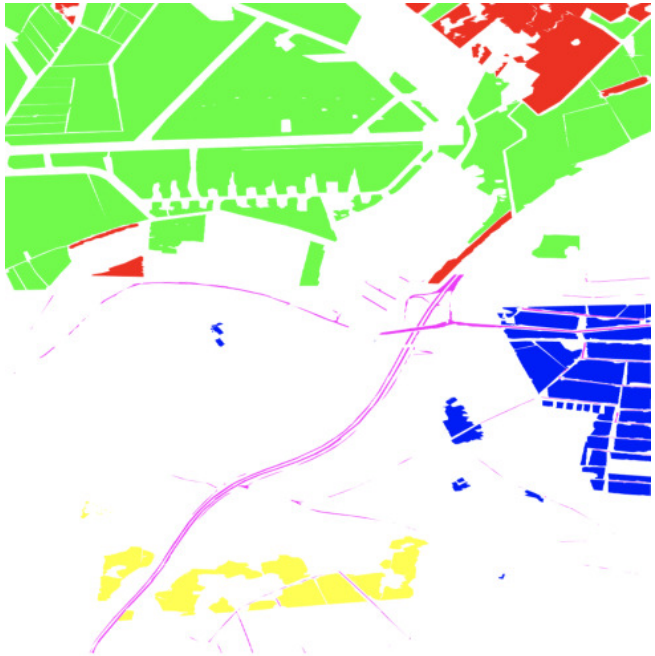
- If for the considered application it is easy to hand-craft class-separable features, no need to learn them
- If it is not easy to discriminate between categories, learning features often helps

How to evaluate performance of a classifier?

- **Common strategy:**
  1. Split all available data into training and test sets
  2. Train a classifier using the training data
  3. Compare the obtained results with the test data



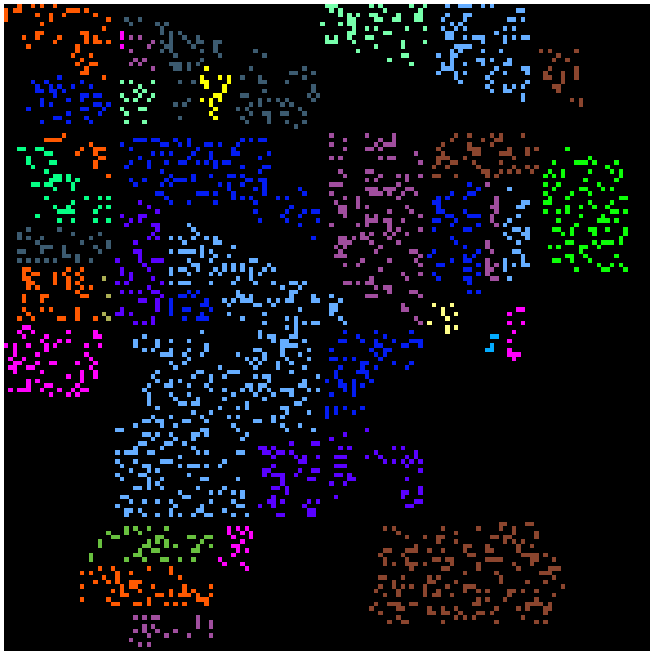
Training set



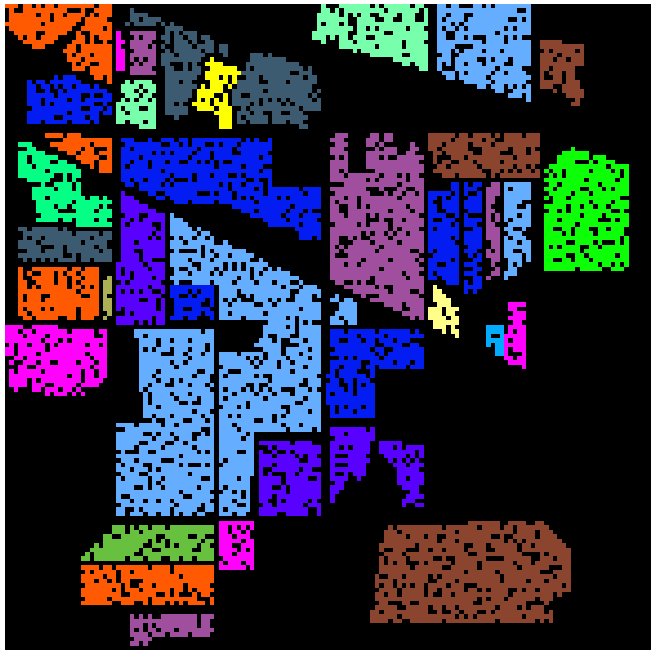
Test set

How to evaluate performance of a classifier?

- **Common strategy:**
  1. Split all available data into training and test sets
  2. Train a classifier using the training data
  3. Compare the obtained results with the test data



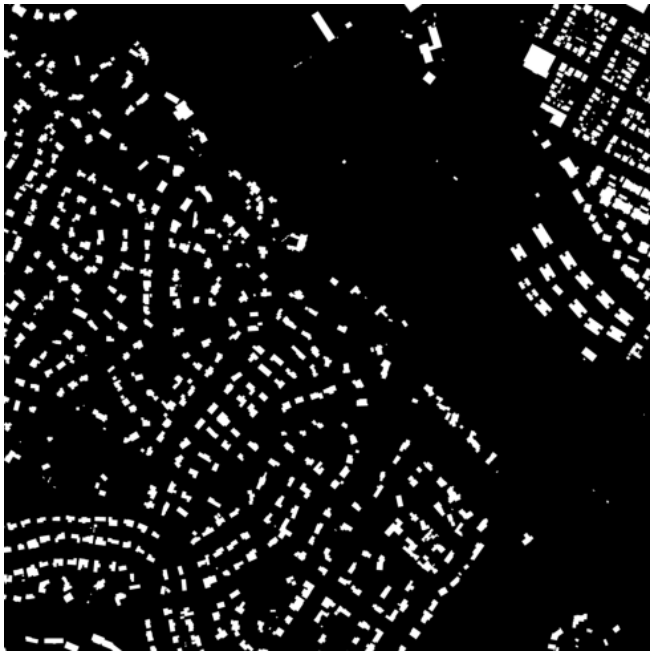
Training set



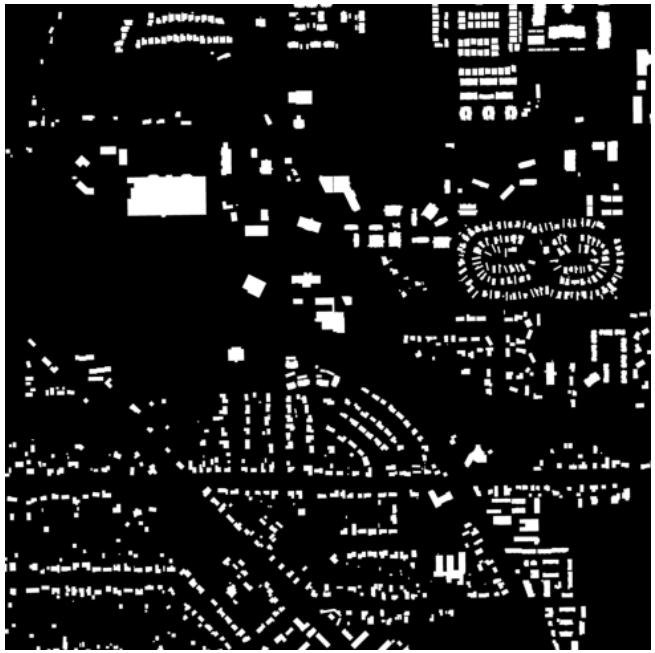
Test set

How to evaluate performance of a classifier?

- **Common strategy:**
  1. Split all available data into training and test sets
  2. Train a classifier using the training data
  3. Compare the obtained results with the test data



Training set



Test set

How to compare the obtained results with the test data?

1. **Visual comparison** of classification maps

How to compare the obtained results with the test data?

1. **Visual comparison** of classification maps
   - Very important! Reveals weaknesses of the classifier
2. **Accuracy measures**: overall, class-specific, average, kappa coefficient, intersection over union

How to compare the obtained results with the test data?

- **Confusion matrix**: table, where each column represents the instances in a predicted class, while each row represents the instances in an actual class
  - Easy to see where the system mislabels one class as another

Percentage	Classification data				
Reference data	C_1	C_2	C_3	Row total	Class-specific accuracy
C_1	C_{11}	C_{12}	C_{13}	$\sum_i^K C_{1i}$	$\frac{C_{11}}{\sum_i^K C_{1i}}$
C_2	C_{21}	C_{22}	C_{23}	$\sum_i^K C_{2i}$	$\frac{C_{22}}{\sum_i^K C_{2i}}$
C_3	C_{31}	C_{22}	C_{33}	$\sum_i^K C_{3i}$	$\frac{C_{33}}{\sum_i^K C_{3i}}$
Column total	$\sum_i^K C_{i1}$	$\sum_i^K C_{i2}$	$\sum_i^K C_{i3}$	N	
User's accuracy	$\frac{C_{11}}{\sum_i^K C_{i1}}$	$\frac{C_{11}}{\sum_i^K C_{i2}}$	$\frac{C_{33}}{\sum_i^K C_{i3}}$		

  - C_i represents the class i and C_{ij} is the number of pixels classified to the class j and referenced as the class i

How to compare the obtained results with the test data?

Percentage	Classification data				
Reference data	C_1	C_2	C_3	Row total	Class-specific accuracy
C_1	C_{11}	C_{12}	C_{13}	$\sum_i^K C_{1i}$	$\frac{C_{11}}{\sum_i^K C_{1i}}$
C_2	C_{21}	C_{22}	C_{23}	$\sum_i^K C_{2i}$	$\frac{C_{22}}{\sum_i^K C_{2i}}$
C_3	C_{31}	C_{22}	C_{33}	$\sum_i^K C_{3i}$	$\frac{C_{33}}{\sum_i^K C_{3i}}$
Column total	$\sum_i^K C_{i1}$	$\sum_i^K C_{i2}$	$\sum_i^K C_{i3}$	N	
User's accuracy	$\frac{C_{11}}{\sum_i^K C_{i1}}$	$\frac{C_{11}}{\sum_i^K C_{i2}}$	$\frac{C_{33}}{\sum_i^K C_{i3}}$		

- **Overall Accuracy** (OA) is the percentage of correctly classified pixels (K is the number of classes):

$$OA = \frac{\sum_i^K C_{ii}}{\sum_{ij}^K C_{ij}} \times 100\%$$

How to compare the obtained results with the test data?

Percentage	Classification data				
Reference data	C_1	C_2	C_3	Row total	Class-specific accuracy
C_1	C_{11}	C_{12}	C_{13}	$\sum_i^K C_{1i}$	$\frac{C_{11}}{\sum_i^K C_{1i}}$
C_2	C_{21}	C_{22}	C_{23}	$\sum_i^K C_{2i}$	$\frac{C_{22}}{\sum_i^K C_{2i}}$
C_3	C_{31}	C_{22}	C_{33}	$\sum_i^K C_{3i}$	$\frac{C_{33}}{\sum_i^K C_{3i}}$
Column total	$\sum_i^K C_{i1}$	$\sum_i^K C_{i2}$	$\sum_i^K C_{i3}$	N	
User's accuracy	$\frac{C_{11}}{\sum_i^K C_{i1}}$	$\frac{C_{11}}{\sum_i^K C_{i2}}$	$\frac{C_{33}}{\sum_i^K C_{i3}}$		

- **Class Accuracy** (CA, or producer's accuracy) is the percentage of correctly classified pixels for a given class:

$$CA_i = \frac{C_{ii}}{\sum_j^K C_{ij}} \times 100\%$$

How to compare the obtained results with the test data?

Percentage	Classification data				
Reference data	C_1	C_2	C_3	Row total	Class-specific accuracy
C_1	C_{11}	C_{12}	C_{13}	$\sum_i^K C_{1i}$	$\frac{C_{11}}{\sum_i^K C_{1i}}$
C_2	C_{21}	C_{22}	C_{23}	$\sum_i^K C_{2i}$	$\frac{C_{22}}{\sum_i^K C_{2i}}$
C_3	C_{31}	C_{22}	C_{33}	$\sum_i^K C_{3i}$	$\frac{C_{33}}{\sum_i^K C_{3i}}$
Column total	$\sum_i^K C_{i1}$	$\sum_i^K C_{i2}$	$\sum_i^K C_{i3}$	N	
User's accuracy	$\frac{C_{11}}{\sum_i^K C_{i1}}$	$\frac{C_{11}}{\sum_i^K C_{i2}}$	$\frac{C_{33}}{\sum_i^K C_{i3}}$		

- **Average Accuracy** (AA) is the mean of class-specific accuracies for all the classes:

$$AA = \frac{\sum_i^K CA_i}{K} \times 100\%$$

How to compare the obtained results with the test data?

- **Kappa Coefficient** (κ) is the percentage of agreement, *i.e.*, correctly classified pixels, corrected by the number of agreements that would be expected purely by chance:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \times 100\%,$$

$$P_o = OA/100\%,$$

$$P_e = \frac{1}{N^2} \sum_i^K C_{i\cdot} C_{\cdot i},$$

$$C_{i\cdot} = \sum_j^K C_{ij}, \quad C_{\cdot i} = \sum_j^K C_{ji},$$

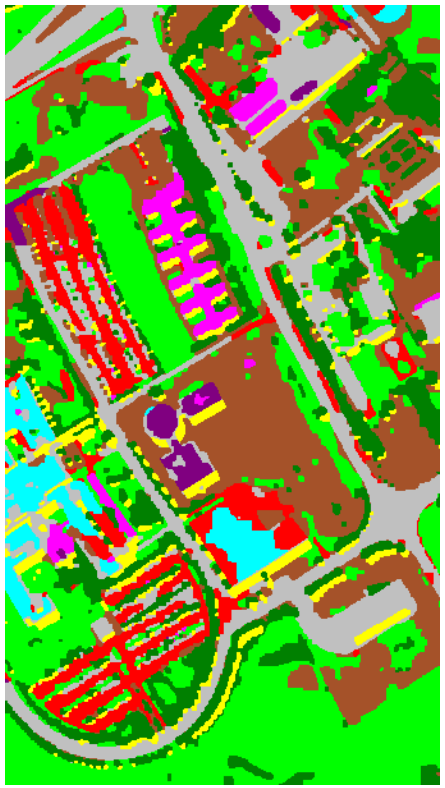
Percentage	Classification data				
Reference data	C_1	C_2	C_3	Row total	Class-specific accuracy
C_1	C_{11}	C_{12}	C_{13}	$\sum_i^K C_{1i}$	$\frac{C_{11}}{\sum_i^K C_{1i}}$
C_2	C_{21}	C_{22}	C_{23}	$\sum_i^K C_{2i}$	$\frac{C_{22}}{\sum_i^K C_{2i}}$
C_3	C_{31}	C_{22}	C_{33}	$\sum_i^K C_{3i}$	$\frac{C_{33}}{\sum_i^K C_{3i}}$
Column total	$\sum_i^K C_{i1}$	$\sum_i^K C_{i2}$	$\sum_i^K C_{i3}$	N	
User's accuracy	$\frac{C_{11}}{\sum_i^K C_{i1}}$	$\frac{C_{11}}{\sum_i^K C_{i2}}$	$\frac{C_{33}}{\sum_i^K C_{i3}}$		

where N is the number of referenced pixels

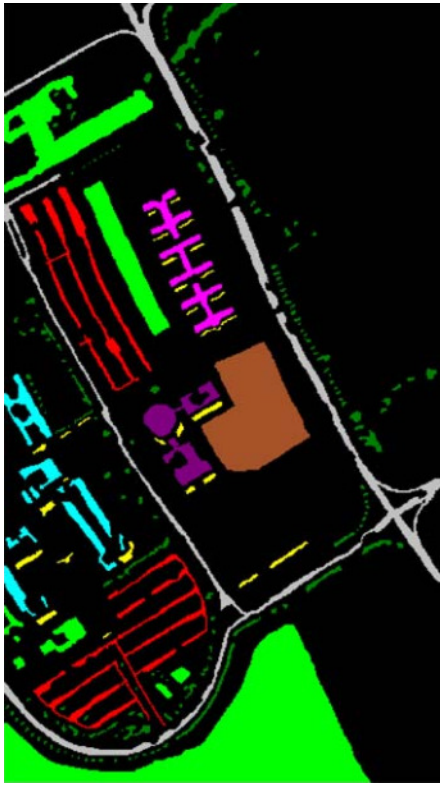
How to compare the obtained results with the test data?

Input image

Classification map



Ground-truth

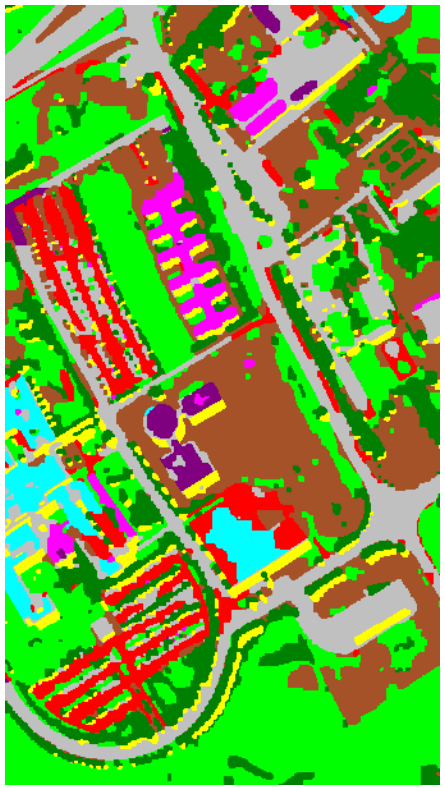


- Overall accuracy $> 90\%$, average accuracy $> 90\%$
- Is this classification result satisfactory?

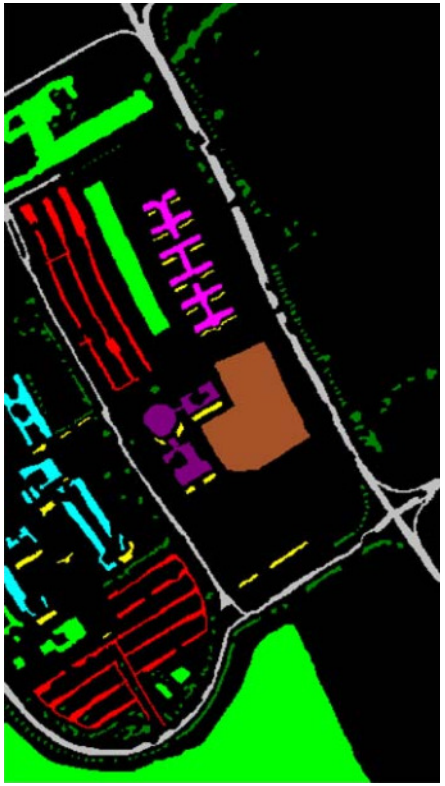
How to compare the obtained results with the test data?

Input image

Classification map



Ground-truth



- Overall accuracy $> 90\%$, average accuracy $> 90\%$
- Is this classification result satisfactory? $\Rightarrow$ Depends on the application

Criterion for object-based classification?

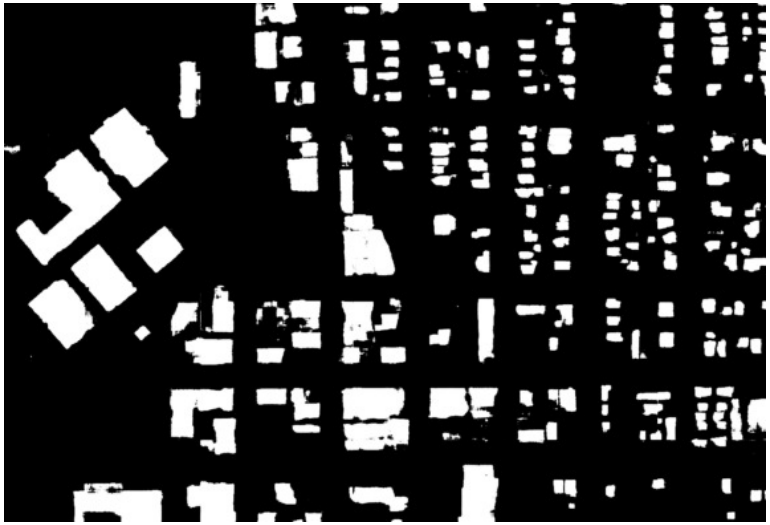
- Intersection over union (IoU) criterion
  - Number of pixels assigned to a particular class both in the classified image and in the ground truth, divided by the total amount of pixels labeled as such in either of them
  - Object-based overlap measure typically used for imbalanced datasets

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Criterion for object-based classification?

- Intersection over union (IoU) criterion
  - Number of pixels assigned to a particular class both in the classified image and in the ground truth, divided by the total amount of pixels labeled as such in either of them
  - Object-based overlap measure typically used for imbalanced datasets

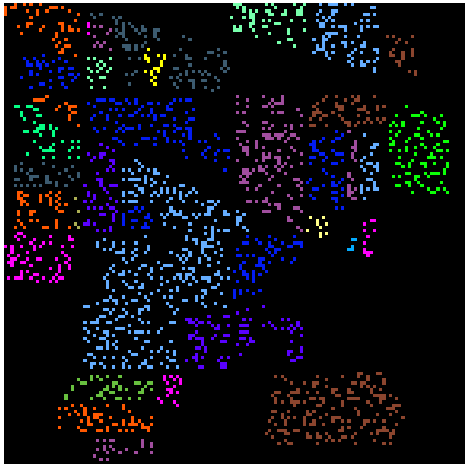
Color image



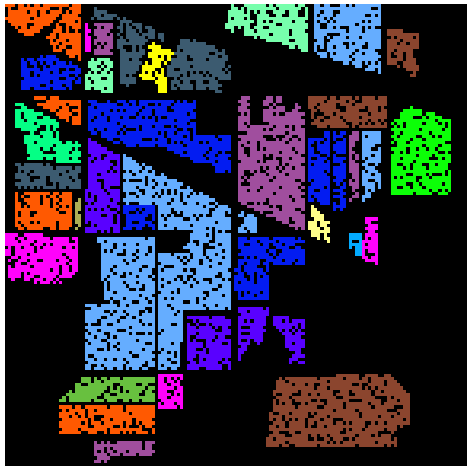
OA = 94%, IoU = 61%

How to split data into training and test sets?

- Sampling strategies
  - Train on one area, test on another area
  - Random sampling: randomly select training samples within the area of each class
  - Patch sampling: image is divided into blocks, test samples are from blocks that haven't been used for training
  - Cluster sampling



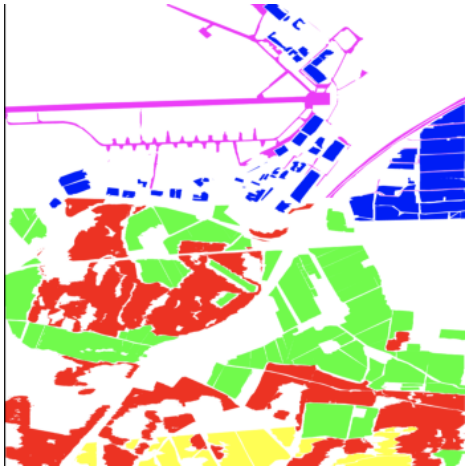
Training set



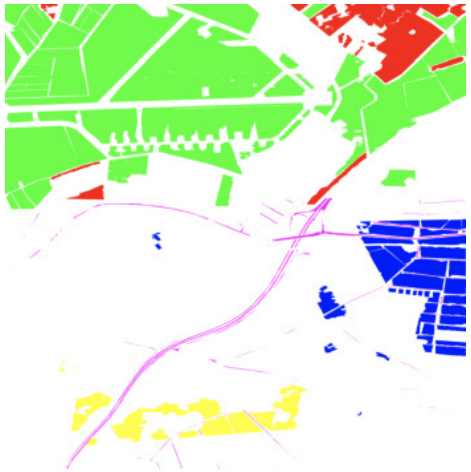
Test set

How to split data into training and test sets?

- Sampling strategies
  - Train on one area, test on another area
  - Random sampling: randomly select training samples within the area of each class
  - Patch sampling: image is divided into blocks, test samples are from blocks that haven't been used for training
  - Cluster sampling



Training set



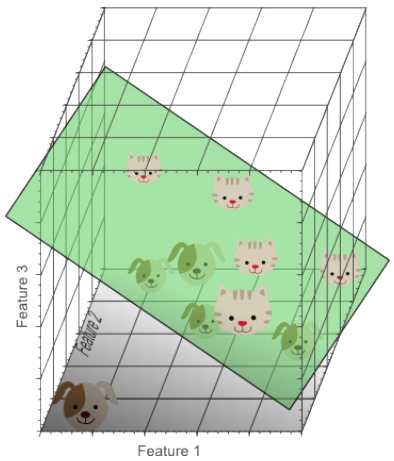
Test set

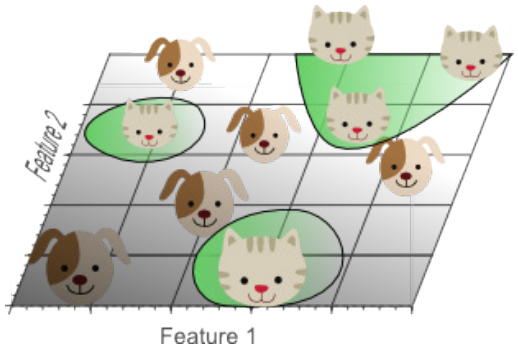
Choose parameters for a classifier?

- **Validation:** if a lot of training data, we can use $X\%$ for training and $Y\%$ for validaton
  - Test on different parameters
  - Retain parameters that give the highest accuracy on validation set
- If limited training data $\Rightarrow$ **cross-validation**
  - Different types
  - Popular choice: k-fold cross-validation
    - Randomly partition the training data into k equal sized subsets
    - k times: one of k subsets is used for validation, and the rest of data are used for training

Underfitting versus overfitting

- **Underfitting:** machine learning algorithm cannot capture the underlying trend of the data
  - Often a result of an excessively simple model
- **Overfitting:** Model or the algorithm fits the data too well & captures the noise of the data
  - Often a result of an excessively complicated model
  - Can be prevented by fitting multiple models and using validation or cross-validation



⇐ Good fit

Overfitting ⇒