>>> Feature Extraction
>>> GRSS Summer School

Name: Mathieu Fauvel (UMR Dynafor)
Date: *[2017-04-26 Wed 10:30]–[2017-04-26 Wed 12:00]*

>>> Outline

>>> Illustration

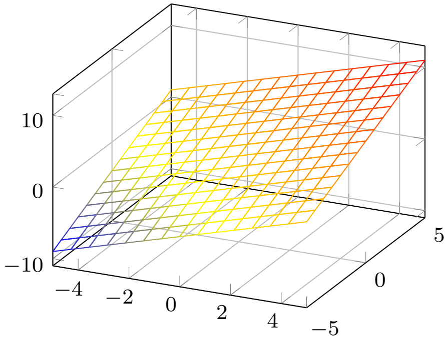
⋆ Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

High dimensional saces are mostly empty !

- ⋆ Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

High dimensional saces are mostly empty !

- ⋆ Multivariate data live in a lower dimensional space



- Feature extraction is important in remote sensing because:
  - It reduces the size of the data,
  - It limits the spatial and spectral redundancy,
  - It permits visualization of the data,
  - It mitigates the *curse of dimensionality*.
- Extraction techniques:
  - Spectral
    - Physically based method,
    - Statistical methods.
  - Spatial:
    - Linear filters,
    - Non linear techniques (Mathematical Morphology)

>>> Outline

- Spectral indices are a linear/non-linear combination of two (or more) spectral bands.
- They provides information as a *single number* about:
  - Plant structure,
  - Biochemistry,
  - Humidity,
  - Stress.
- Four main types [TLH11]:

Name	Formulae
Difference vegetation index	$R_{\lambda_1} - R_{\lambda_2}$
Ratio vegetation index	$\frac{R_{\lambda_1}}{R_{\lambda_2}}$
Normalized difference vegetation index	$\frac{R_{\lambda_1} - R_{\lambda_2}}{R_{\lambda_1} + R_{\lambda_2}}$
Soil-adjusted vegetation index	$(1+L) \times \frac{R_{\lambda_1} - R_{\lambda_2}}{R_{\lambda_1} - R_{\lambda_2} + L}$

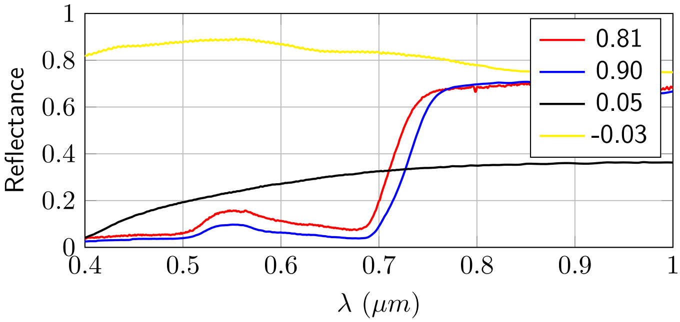
- *The three last indexes are invariant to a multiplicative factor*

Index database : `http://www.indexdatabase.de/`

Name	Formulae (λ nm)
Normalized Difference Vnegetation index	$\frac{R_{\lambda_{800}} - R_{\lambda_{670}}}{R_{\lambda_{800}} + R_{\lambda_{670}}}$
Modified Soil-Adjusted Vegetation Index	$\frac{1}{2}\left[2R_{\lambda_{800}} + 1 - \sqrt{(2R_{\lambda_{800}} + 1)^2 - 8(R_{\lambda_{800}} - R_{\lambda_{670}})}\right]$
Modified Chlorophyll Absorption Ratio Index	$[(R_{\lambda_{700}} - R_{\lambda_{670}}) - 0.2(R_{\lambda_{700}} - R_{\lambda_{550}})] \times \frac{R_{\lambda_{700}}}{R_{\lambda_{670}}}$
Normalized Difference Water Index	$\frac{R_{\lambda_{858}} - R_{\lambda_{1240}}}{R_{\lambda_{858}} + R_{\lambda_{1240}}}$
Datt Reflectance Index	$\frac{R_{\lambda_{816}} - R_{\lambda_{2218}}}{R_{\lambda_{816}} + R_{\lambda_{2218}}}$
Normalized Difference Redness Index	$\frac{R_{\lambda_{540}} - R_{\lambda_{700}}}{R_{\lambda_{540}} + R_{\lambda_{700}}}$
Soil Brightness Index	$0.406R_{\lambda 550} + 0.600R_{\lambda 650} + 0.645R_{\lambda 750} + 0.243R_{\lambda 950}$

$$\text{NDVI} = \frac{R_{\lambda_{800}} - R_{\lambda_{670}}}{R_{\lambda_{800}} + R_{\lambda_{670}}}$$

- $-1 \leq$ NVDI ≤ 1
- NDVI < 0: surfaces other thatn plant cover
- NDVI ≈ 0: bare soil
- NDVI ≥ 0.1: vegetation cover (higher values correspond to more dense covers)

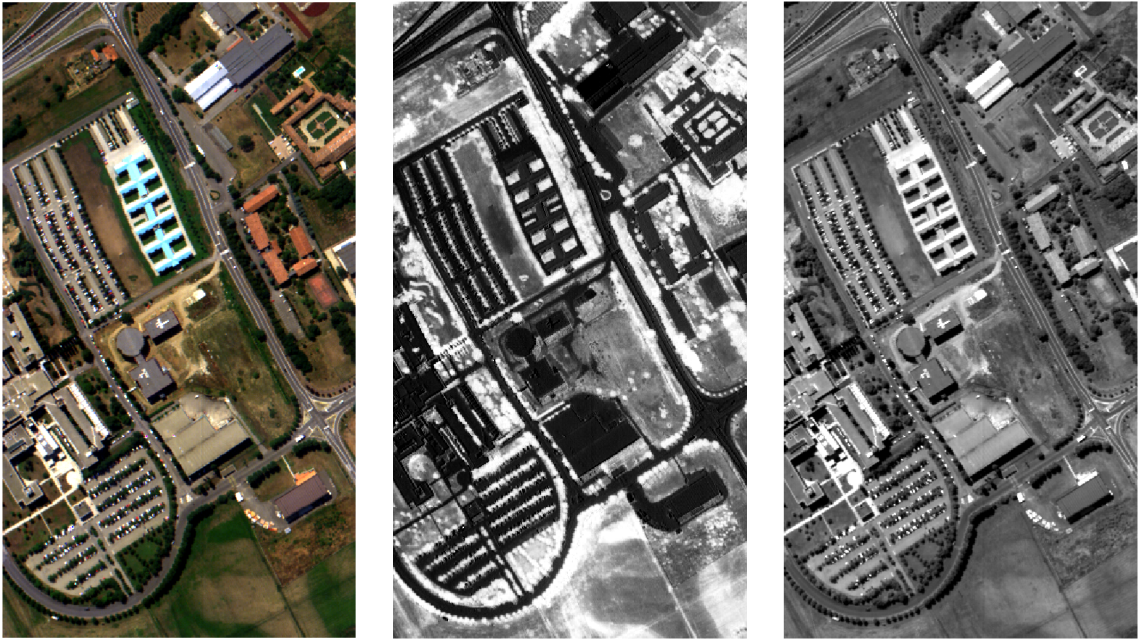


- Peri-urban area
- Rosis-3 sensor
- 103 Spectral bands (400nm-900nm)
- 1.5 meter per pixel spatial resolution
- 610 $\times$ 340 pixels

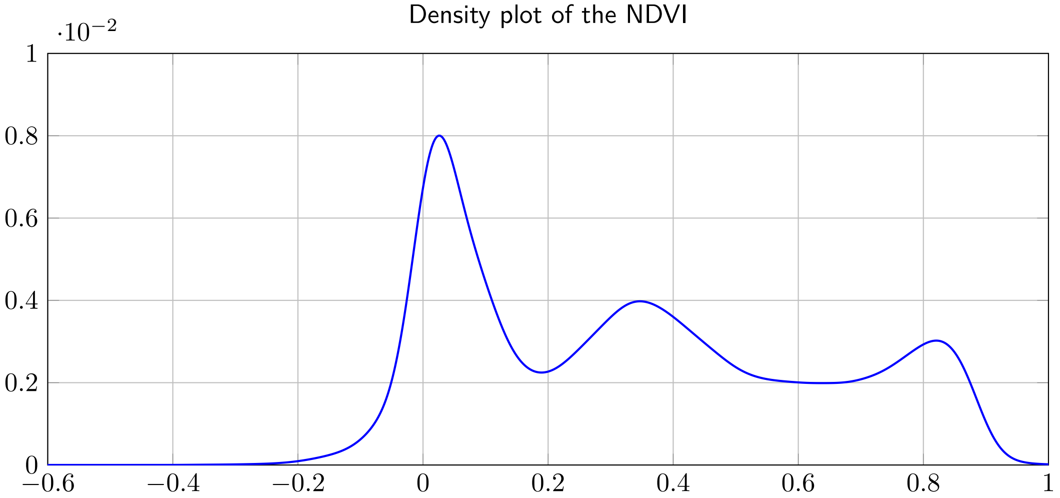
- OTB is a C++ library for remote sensing images processing.
- It is free, open-source and available for most OS (window, apple, linux)
- OTB-Applications are set of tools appropriated for big/large images
- They are avalaible from QGIS, Python and Bash
- To compute the NDVI

```
# Computation of the NDVI
otbcli_BandMath -il ../Data/university.tif -out ../Data/university_ndvi.tif \
-exp "(im1b83-im1b56)/(im1b83+im1b56)"

# Computation of the SBI
otbcli_BandMath -il ../Data/university.tif -out ../Data/university_sbi.tif \
-exp "0.406*im1b31 + 0.6*im1b52 + 0.645*im1b73"
```

>>> University of Pavia - Spectral Indices

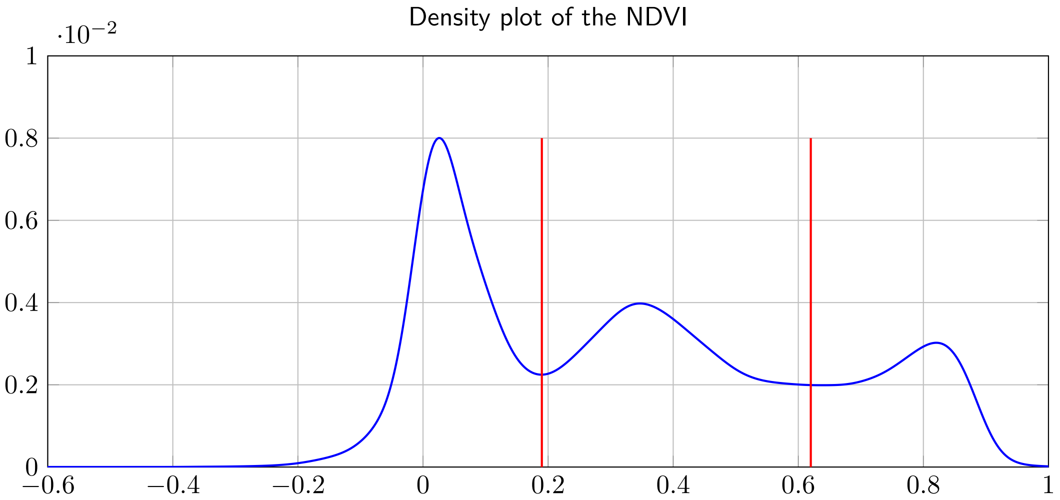


>>> Where is the vegetation 1/2 ?




```
# Segmentation of the NDVI in three classes
otbcli_BandMath -il ../Data/university_ndvi.tif -out ../Data/university_ndvi_segmented.tif \
-exp "(im1b1<0.19?1:(im1b1<0.62?2:3))"
```

>>> Where is the vegetation 1/2 ?

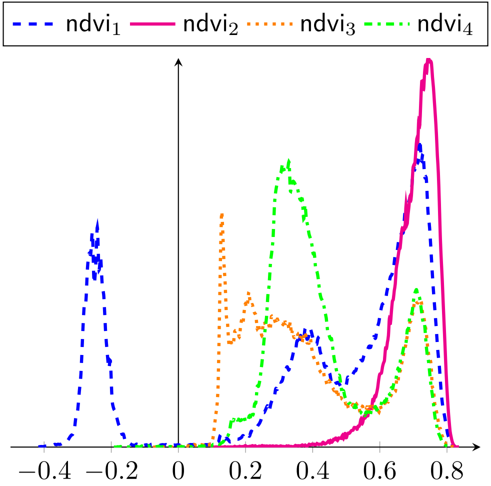



```
# Segmentation of the NDVI in three classes
otbcli_BandMath -il ../Data/university_ndvi.tif -out ../Data/university_ndvi_segmented.tif \
-exp "(im1b1<0.19?1:(im1b1<0.62?2:3))"
```

>>> Where is the vegetation 2/2 ?

>>> Could you find the good one ?

Image



NDVI Histogram

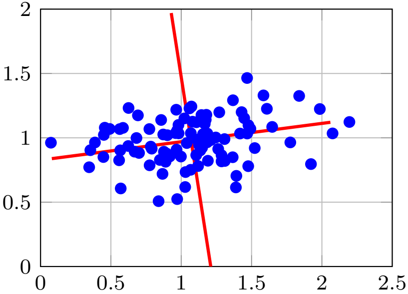
From the histogram, which one does correspond to the NDVI of the image ?

>>> Outline

- Linear transformation used to reduce the dimensionality of the data [Jol02].

$$z_i = \langle \mathbf{v}_i, \mathbf{x} \rangle$$

- Find features $\mathbf{z}$ that account for most of the variability of the data:
  - $z_1, \ z_2, \ z_3, \ldots$ are mutually uncorrelated,
  - var(z_i) is as large as possible,
  - var$(z_1) >$ var$(z_2) >$ var$(z_3) > \ldots$



>>> Maximization of the variance 1/2

- Search $\mathbf{v}_1$ such as $\max \mathsf{var}(z_1)$:

$$\begin{aligned} \mathsf{var}(z_1) &= \mathsf{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Search $\mathbf{v}_1$ such as $\max \mathsf{var}(z_1)$:

$$\begin{aligned}\mathsf{var}(z_1) &= \mathsf{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_1\end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$

>>> Maximization of the variance 1/2

- Search $\mathbf{v}_1$ such as $\max \mathsf{var}(z_1)$:

$$\begin{aligned} \mathsf{var}(z_1) &= \mathsf{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha \hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1 (1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

>>> Maximization of the variance 1/2

- ⋆ Search $\mathbf{v}_1$ such as $\max \mathsf{var}(z_1)$:

$$\begin{aligned}\mathsf{var}(z_1) &= \mathsf{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1\end{aligned}$$

- ⋆ Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- ⋆ Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1(1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- ⋆ Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\boldsymbol{\Sigma}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- Search $\mathbf{v}_1$ such as $\max \text{var}(z_1)$:

$$\begin{aligned} \text{var}(z_1) &= \text{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha \hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_1 + \lambda_1 (1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\mathbf{\Sigma} \mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is an eigenvector of the covariance matrix of $\mathbf{x}$:

$$\mathbf{\Sigma} \mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

- Search $\mathbf{v}_1$ such as $\max \text{var}(z_1)$:

$$\begin{aligned}\text{var}(z_1) &= \text{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1\end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1(1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\boldsymbol{\Sigma}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is an eigenvector of the covariance matrix of $\mathbf{x}$:

$$\boldsymbol{\Sigma}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is the eigenvector corresponding to the largest eigenvalues !

$$\text{var}(z_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 = \lambda_1 \mathbf{v}_1^\top \mathbf{v}_1 = \lambda_1$$

>>> Maximization of the variance 2/2

- ⋆ Search $\mathbf{v}_2$ such as $\max \mathsf{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$

- Search $\mathbf{v}_2$ such as $\max \mathsf{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_1(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_1 (1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1 (0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_1 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_1(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_1 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_1 (1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1 (0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_1 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Hence, we have

$$\mathbf{\Sigma}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

- ⋆ Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- ⋆ Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_1 (1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1 (0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- ⋆ Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_1 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- ⋆ At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- ⋆ Hence, we have

$$\mathbf{\Sigma}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

- ⋆ $\mathbf{v}_2$ is the eigenvector corresponding the *second largest* eigenvalues

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \boldsymbol{\Sigma} \mathbf{v}_2 + \lambda_1 (1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1 (0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\boldsymbol{\Sigma}\mathbf{v}_2 - 2\lambda_1 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \boldsymbol{\Sigma}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Hence, we have

$$\boldsymbol{\Sigma}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

- $\mathbf{v}_2$ is the eigenvector corresponding the *second largest* eigenvalues
- $\mathbf{v}_k$ is the eigenvector corresponding the k^{th} *largest* eigenvalues

1. Empirical estimation the mean value:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

2. Empirical estimation the covariance matrix:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top$$

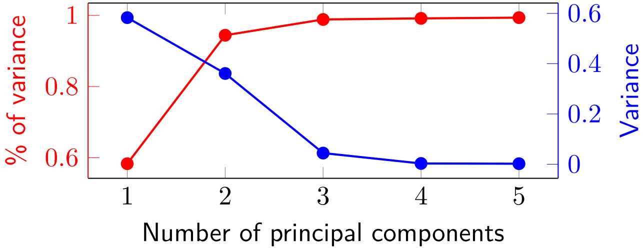
3. Compute p first eigenvalues/eigenvectors... How to choose p ? Explained variance:

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

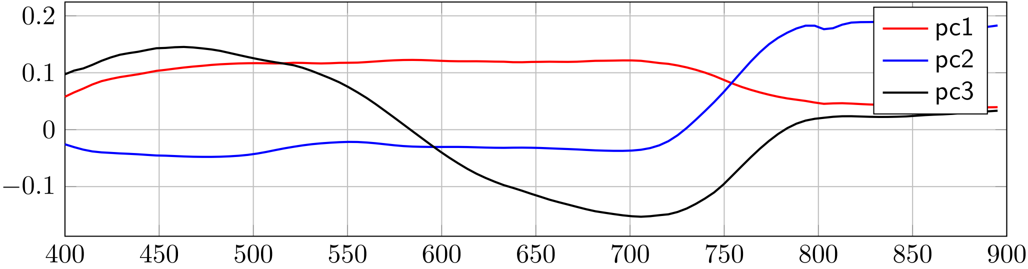
4. Tips for high dimensional data set: if $n < d$ see [MLC16] page 420

```
import rasterTools as rt
import scipy as sp
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load data set
im,GeoT,Proj = rt.open_data('../Data/university.tif')
[h,w,b]=im.shape
im.shape=(h*w,b)
wave = sp.loadtxt('../Data/waves.csv',delimiter=',')

# Do PCA
pca = PCA()
pca.fit(im)

# Plot explained variance
l = pca.explained_variance_ratio_
print l[:5]
print (l.cumsum()/l.sum())[:5]

# Save Eigenvectors
D = sp.concatenate((wave[:,sp.newaxis],pca.components_[:3,:].T),axis=1)
sp.savetxt('../FeatureExtraction/figures/pca_pcs.csv',D,delimiter=',')
```

- ⋆ Explained variance



- ⋆ Principal components



```
# Projection of the first PCs
imp = sp.dot(im,pca.components_[:3,:].T)
imp.shape = (h,w,3)

# Save image
rt.write_data('../Data/pca_university.tif',imp,GeoT,Proj)
```

>>> Kernel PCA

- PCA is limited to second order information
- To capture higher-order statistics, it is possible to map the data onto another space $\mathcal{H}$

$$\begin{aligned}\phi : \mathbb{R}^d &\rightarrow \mathcal{H} \\ \mathbf{x} &\mapsto \phi(\mathbf{x}).\end{aligned}$$

- In $\mathcal{H}$, conventional PCA can be applied.
- Using the *kernel trick* it is possible to directly work on the *kernel matrix* in $\mathbb{R}^d$

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}.$$

>>> Kernel PCA

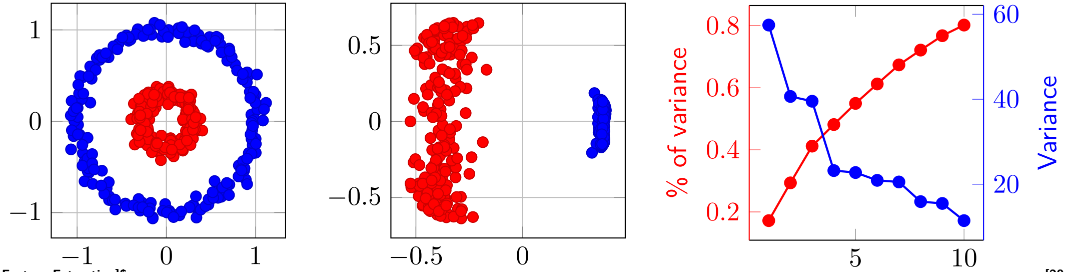
- ⋆ PCA is limited to second order information
- ⋆ To capture higher-order statistics, it is possible to map the data onto another space $\mathcal{H}$

$$\begin{aligned} \phi : \mathbb{R}^d &\rightarrow \mathcal{H} \\ \mathbf{x} &\mapsto \phi(\mathbf{x}). \end{aligned}$$

- ⋆ In $\mathcal{H}$, conventional PCA can be applied.
- ⋆ Using the *kernel trick* it is possible to directly work on the *kernel matrix* in $\mathbb{R}^d$

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}.$$

- ⋆ KPCA versus PCA:



- Choose the kernel and its parameters
- Compute the kernel matrix $\mathbf{K}$ for all the pixels (or a subset)
- Center the matrix

$$\mathbf{K}_c = \mathbf{K} - \mathbf{1}_n\mathbf{K} - \mathbf{K}\mathbf{1}_n + \mathbf{1}_n\mathbf{K}\mathbf{1}_n$$

- Solve the eigenproblems

$$\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha} \text{ subject to } \|\boldsymbol{\alpha}\|_2 = \frac{1}{\lambda}$$

- Project on the p first *kernel principal components*: $\phi^{kpc}(\mathbf{x}) = \begin{bmatrix}\phi_1^{kpc}(\mathbf{x}) & \dots & \phi_p^{kpc}(\mathbf{x})\end{bmatrix}^t$

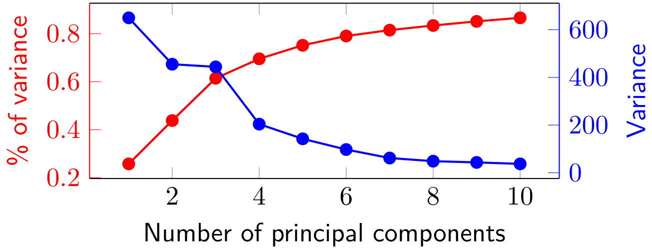
$$\phi_j^{kpc}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_{ki} k(\mathbf{x}_i, \mathbf{x})$$

From [FCB09].

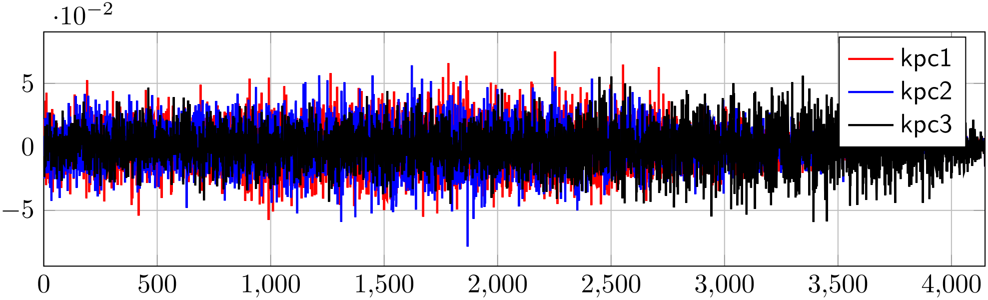
```
import rasterTools as rt
import scipy as sp
from sklearn.decomposition import KernelPCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Load data set
im,GeoT,Proj = rt.open_data('../Data/university.tif')
[h,w,b]=im.shape
im.shape=(h*w,b)
wave = sp.loadtxt('../Data/waves.csv',delimiter=',')

# Scale data
sc = StandardScaler()
im = sc.fit_transform(im)

# Do KPCA
kpca = KernelPCA(kernel='rbf',gamma=1.0/b,n_jobs=-1)
kpca.fit(im[::50,:]) # Use a subset of the total number of pixels
```

>>> KPCA case study 2/3

- ⋆ Explained variance



- ⋆ Principal components

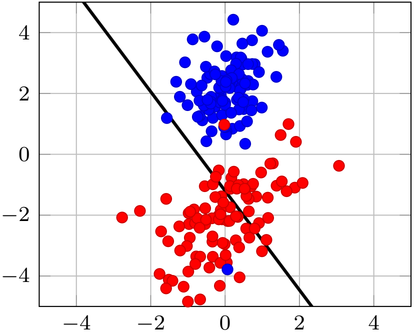


>>> KPCA case study 3/3

```
imp = kpca.transform(im)[:,:3]
imp.shape = (h,w,3)

# Save image
rt.write_data('../Data/kpca_university.tif',imp,GeoT,Proj)
```



- We observe some $\{\mathbf{x}_i, y_i\}_{i=1}^n$
- Use the label information to find the linear features that highlight differences among classes



- FDA: find $\mathbf{a}$ such as the ratio between the *between projected variance* and the *sample projected variance* is maximal

- Between-class covariance matrix:

$$\mathbf{B} = \frac{1}{n}\sum_{c=1}^{C} n_c(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\top$$

- Sample covariance matrix

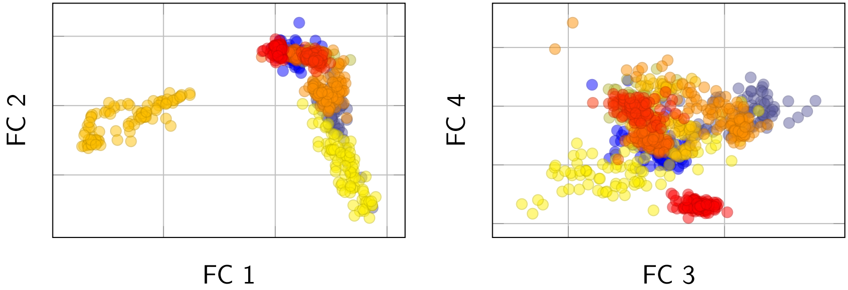
$$\boldsymbol{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top$$

- The Fisher discriminant subspace is given by the eigenvectors of $\boldsymbol{\Sigma}^{(-1)}\mathbf{B}$
- Remark: there are at most $C-1$ eigenvectors because Rank$(\mathbf{B}) \leq C-1$.

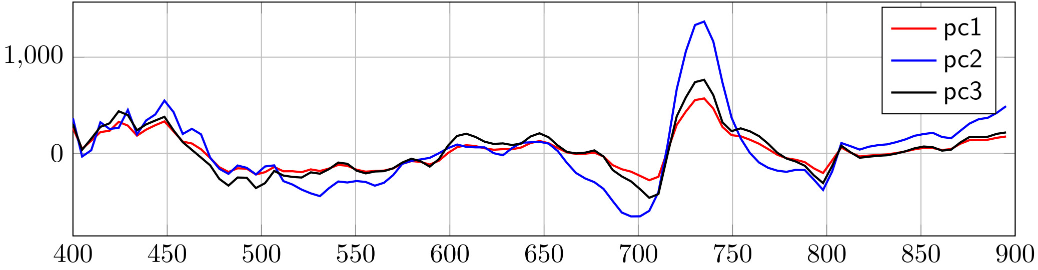
>>> FDA case study 1/3

```
import rasterTools as rt
import scipy as sp
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Load data set
X,y=rt.get_samples_from_roi('../Data/university.tif','../Data/university_gt.tif')
wave = sp.loadtxt('../Data/waves.csv',delimiter=',')

# Select the same number of samples
nt = 900
xt,yt=[],[]
for i in sp.unique(y):
    t = sp.where(y==i)[0]
    nc = t.size
    rp =  sp.random.permutation(nc)
    xt.extend(X[t[rp[0:nt]],:])
    yt.extend(y[t[rp[0:nt]]])

xt = sp.asarray(xt)
yt = sp.asarray(yt)

# Do LDA
lda = LinearDiscriminantAnalysis(solver='eigen',shrinkage='auto')
lda.fit(xt,yt.ravel())
```

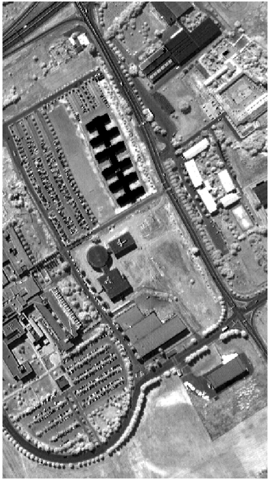
- Projection on Fisher components



- Fisher components



```
im,GeoT,Proj = rt.open_data('../Data/university.tif')
[h,w,b]=im.shape
im.shape=(h*w,b)
imp = lda.transform(im)[:,:3]
imp.shape = (h,w,3)
# Save image
rt.write_data('../Data/lda_university.tif',imp,GeoT,Proj)
```



- Feature selection: pick few features *from* the original ones (no combination)
- In general, for feature selection, we need:
  - *Criterion* to evaluate how perform the model with a given subset
  - *Optimization procedure* to find the subset that minimizes/maximizes the criterion
- For instance:

Criterion	Optimization	Ref.
Entropy	Genetic algorithm	[Cha07]
Jeffries Matusita	Exhaustive Search	[SM07]
Classification error	Forward search/GA	[CBN14; LFG17]
ℓ_1 norm	Linear-SVM	[Tui+14]

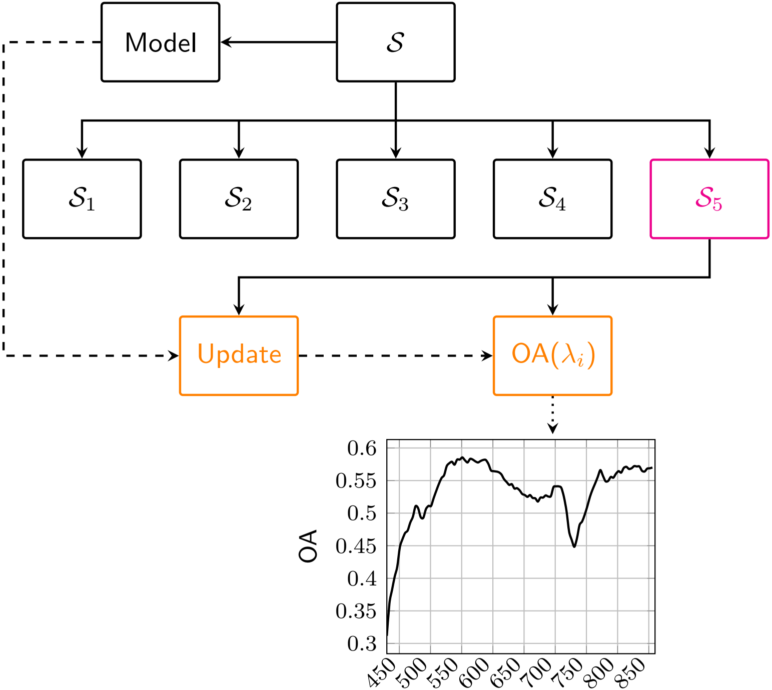
- Fast forward strategy based on a nonlinear model driven by an estimate of the classification error or a measure of separability:

Criterion	Type	Complexity
Overall accuracy	Accuracy	High
Cohen's kappa	Accuracy	High
F1 mean	Accuracy	High
Kullback-Leibler divergences	Divergence	Low
Jeffries-Matusita distance	Divergence	Low

- Use *Gaussian Mixture Models* (natural extension du multiclass problem)
- Fast update and fast forward search [LFG17]: based on linear algebra

The forward feature selection works as follow:

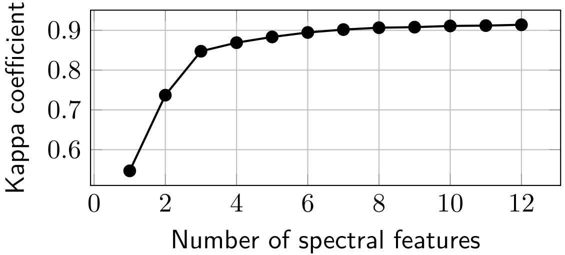
- ⋆ Starts with an empty pool F of selected features,
- ⋆ Select the feature f_1 that provides the best value for the selected criterion and add it to F.
- ⋆ Select the feature f_2 such that the couple of features (f_1, f_2) provides the best value for the selected criterion and add it to F.
- ⋆ Select the feature f_3 such that the triplet of features (f_1, f_2, f_3) ...
- ⋆ ...
- ⋆ The algorithm stops either if the increase of the criterion is too low or if the maximum number of features is reached.


Model
$\mathcal{S}$
$\mathcal{S}_1$
$\mathcal{S}_2$
$\mathcal{S}_3$
$\mathcal{S}_4$
$\mathcal{S}_5$
Update
OA(λ_i)
OA
0.6
0.55
0.5
0.45
0.4
0.35
0.3
450
500
550
600
650
700
750
800
850

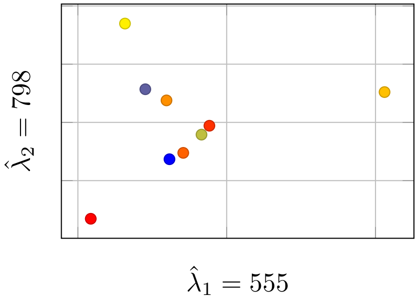
>>> FFFS case study 1/3

```python
import rasterTools as rt
import scipy as sp
import npfs as npfs

# Load data set
X,y=rt.get_samples_from_roi('../Data/university.tif','../Data/university_gt.tif')
wave = sp.loadtxt('../Data/waves.csv',delimiter=',')

# Select the same number of samples
nt = 900
xt,yt=[],[]
for i in sp.unique(y):
    t = sp.where(y==i)[0]
    nc = t.size
    rp =  sp.random.permutation(nc)
    xt.extend(X[t[rp[0:nt]],:])
    yt.extend(y[t[rp[0:nt]]])

xt = sp.asarray(xt)
yt = sp.asarray(yt)

# Do FFFS
maxVar = 12
model = npfs.GMMFeaturesSelection()
model.learn_gmm(xt,yt)
idx, crit, [] = model.selection('forward',xt, yt,criterion='kappa', varNb=maxVar, nfold=5)
```

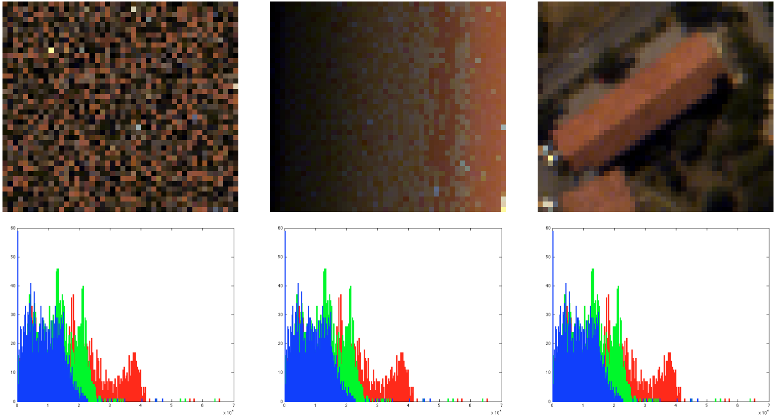
- Criterion



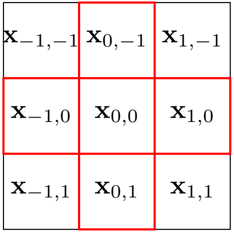
- Mean projection on best bands



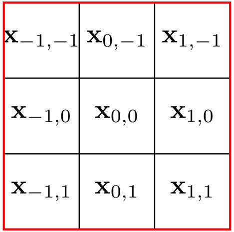
>>> Why spatial feature extraction?



- The neighborhood of a given pixel is the set of pixels that are connected to it.
- For a flat (grayscale) image :



4-connected



8-connected

- Wide range of processing are based on pixel neighborhood
  - De noising,
  - Texture analysis,
  - Edges detection,
  - Pattern recognition,
  - . . .

Steps:

1. Define the template G: 4/8-connected and size
2. Define the processing f on the neighborhood. If f is linear $\leftrightarrow$ convolution.
3. Scan all the pixels:

$$\mathbf{x}_{ij}^{f} = f(\mathbf{x}_1, \ldots, \mathbf{x}_N),\ \mathbf{x}_n \in G(i,j)$$

Max Filter

27	26	25	24	24	23
25	24	23	23	22	21
23	23	22	21	20	20
22	21	20	20	19	18
21	20	19	18	18	17
20	19	18	17	17	16

			23		

- $\star$ $\mathsf{G} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, for a 3×3 neighborhood.
- $\star$ Mean filter

$$\mathbf{x}^m(x, y) = \frac{1}{9} \sum_{i,j=-1}^{1} \mathbf{x}(x + i, y + j)$$

- $\star$ Variance filter:

$$\mathbf{x}^v(x, y) = \frac{1}{9} \sum_{i,j=-1}^{1} \big(\mathbf{x}(x + i, y + j) - \mathbf{x}^m(x, y)\big)^2$$

- $\star$ Range filter:

$$\mathbf{x}^r(x, y) = \max_{i,j \in G} [\mathbf{x}(x + i, y + j)] - \min_{i,j \in G} [\mathbf{x}(x + i, y + j)]$$

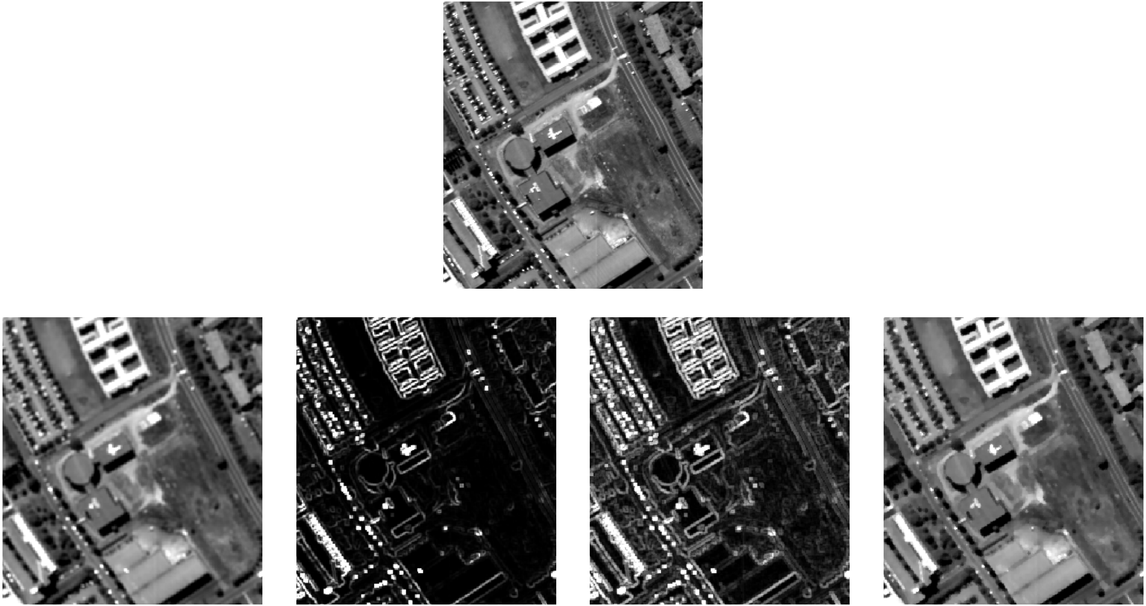
- $\star$ Median filter:

$$\mathbf{x}^m(x, y) = \text{median}_{i,j \in G} [\mathbf{x}(x + i, y + j)]$$

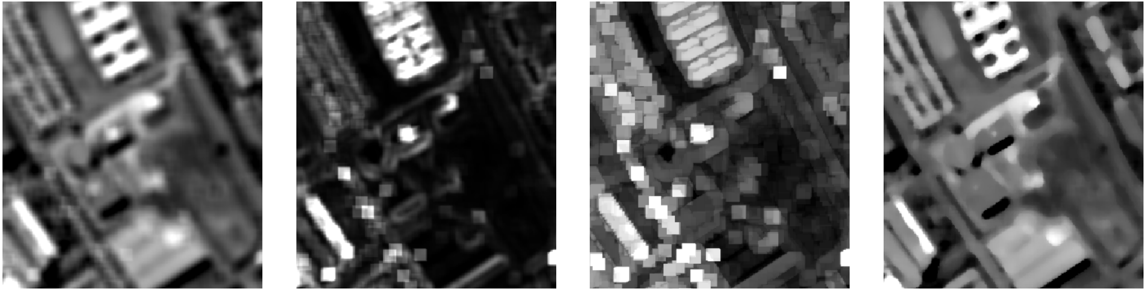
For multidimensional images: Use spectral feature extraction to get flat images! See 3

```
# Compute the different filters with a template of size 3x3 and 11x11
for i in 3 11
do
    # Mean filter
    otbcli_BandMathX -il ../Data/pca_university.tif -out ../Data/pca_mean_${i}_${i}_university.tif \
     -exp "mean(im1b1N${i}x${i}); mean(im1b2N${i}x${i}); mean(im1b3N${i}x${i})"

    # Var filter
    otbcli_BandMathX -il ../Data/pca_university.tif -out ../Data/pca_std_${i}_${i}_university.tif \
     -exp "var(im1b1N${i}x${i}); var(im1b2N${i}x${i}); var(im1b3N${i}x${i})"

    # Range filter
    otbcli_BandMathX -il ../Data/pca_university.tif -out ../Data/pca_range_${i}_${i}_university.tif \
     -exp "vmax(im1b1N${i}x${i})-vmin(im1b1N${i}x${i}); vmax(im1b2N${i}x${i})-vmin(im1b2N${i}x${i});\
                   vmax(im1b3N${i}x${i})-vmin(im1b3N${i}x${i})"

    # Median filter
    otbcli_BandMathX -il ../Data/pca_university.tif -out ../Data/pca_median_${i}_${i}_university.tif \
     -exp "median(im1b1N${i}x${i}); median(im1b2N${i}x${i}); median(im1b3N${i}x${i})"
done
```

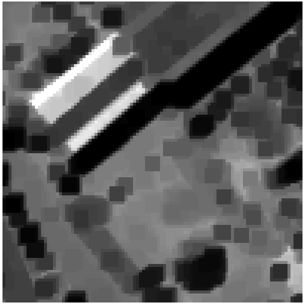
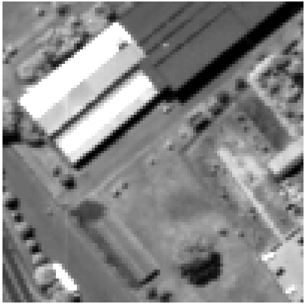
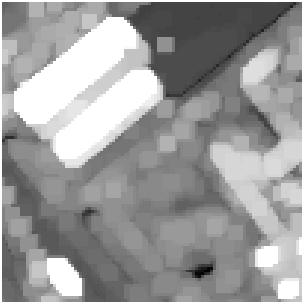
>>> Template filters in action 2/3

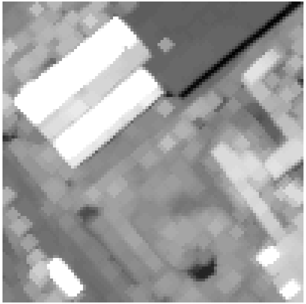


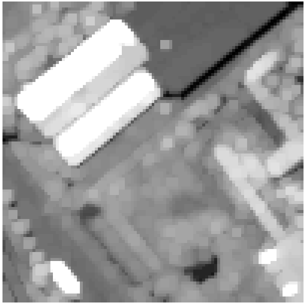
>>> Template filters in action 3/3

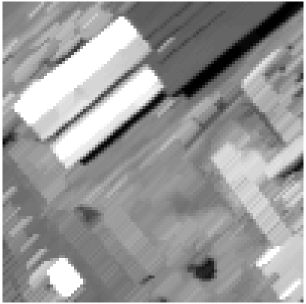


- Mathematical morphology: non-linear image processing.
- A lot of applications in geoscience and remote sensing, see [SP02]
- Erosion: template filter with a $\min$ operation in G (called *structuring element*)
- Dilation: template filter with a $\max$ operation in G







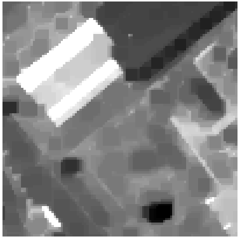
0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

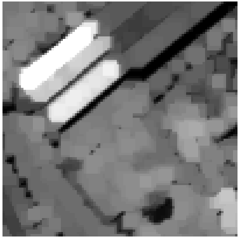
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

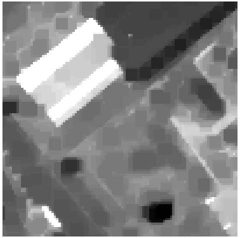
- Opening:
  - *Erosion* followed by a *dilation*
  - Remove bright objects that are smaller than the SE

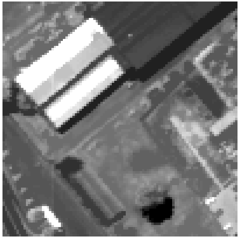
- Closing:
  - *Dilation* followed by an *erosion*
  - Remove dark objects that are smaller than the SE

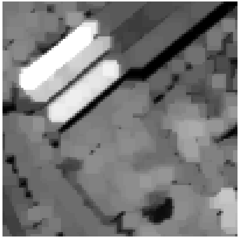


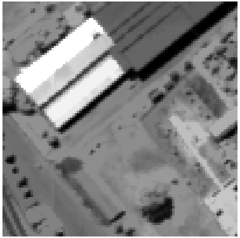


- Opening:
  - *Erosion* followed by a *dilation*
  - Remove bright objects that are smaller than the SE
- Opening by reconstruction:
  - *Erosion* followed by a *reconstruction*
  - *Completely* removes bright objects that are smaller than the SE, otherwise preserve it
- Closing:
  - *Dilation* followed by an *erosion*
  - Remove dark objects that are smaller than the SE
- Closing by reconstruction:
  - *Dilation* followed by an *erosion*
  - *Completely* removes dark objects that are smaller than the SE, otherwise preserve it

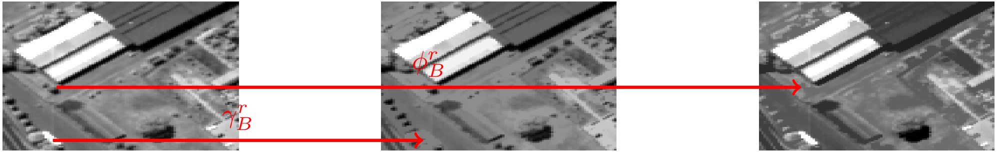








- For a given B, γ_B^r (resp. ϕ_B^r) indicates which clear (dark) objects fit B.



- Applying γ_{B_i} with a set of $\left\{B_i | B_i \subset B_{i+1}, i \in [1, \ldots, n]\right\}$: Opening Profile
- Applying ϕ_{B_i} with a set of $\left\{B_i | B_i \subset B_{i+1}, i \in [1, \ldots, n]\right\}$: Closing Profile

- MM is based on $\inf$ and $\sup$ operators
- No unambiguous \$$\inf$\$/$\sup$ for pixel/vector:

$$\begin{bmatrix}1\\5\\2\end{bmatrix} \overset{?}{\lesseqgtr} \begin{bmatrix}0\\6\\1\end{bmatrix}$$

- Marginal ordering $\Rightarrow$ by band filtering
- Reduced ordering $\Rightarrow h : \mathbb{R}^d \rightarrow \mathbb{R}$

$$\mathbf{x} \mapsto h(x)$$

- Use spectral feature extractio *then* spatial feature extraction.