Machine Learning for Ocean Colour

Classification and Retrieval

Ana B. Ruescas

ana.ruescas@brockmann-consult.de
bruescas@uv.es

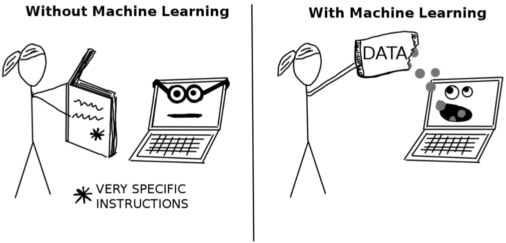
November 4, 2021

Brockmann Consult GmbH
IPL Image Processing Laboratory

Table of Contents

What is Machine Learning?

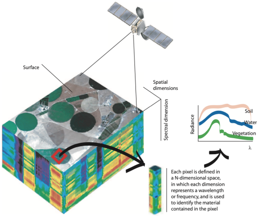
- Machine learning is a set of methods that computers use to make and improve predictions or behaviors based on data.
- The presentation will focus on supervised machine learning: we have a dataset for which we already know the outcome of interest and want to learn to predict the outcome for new data.
- If the output is categorial, the task is called classification, and if it is numerical, it is called regression.
- Clustering tasks (= unsupervised learning) or reinforcement learning are not the object of this lecture.

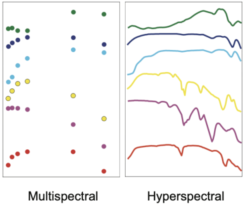


[C. Molnar, Interpretable Machine Learning, 2021]

Satellite image processing

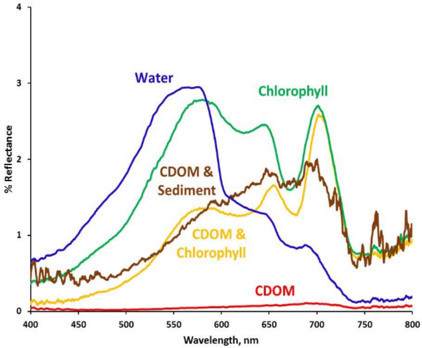
- Materials in a scene reflect, absorb, and emit electromagnetic radiation in a different way depending of their molecular composition and shape
- Remote sensing exploits this physical fact
- Absorption, depth, re-emissions and particular spectral features
- Accurate identification of bio-physical components and processes
- Image spectroscopy allows to identify materials in the scene with unprecedented accuracy



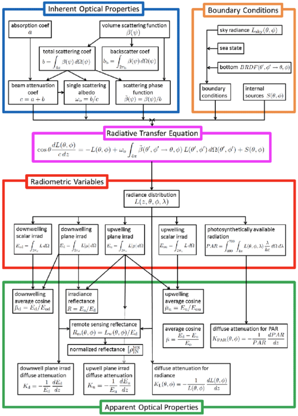


Ocean Colour processing

- Different materials produce different electromagnetic radiation spectra
- The spectrum shows absorptions and emissions at different wavelengths
- The spectral information contained in an image pixel can therefore indicate the various components in the water



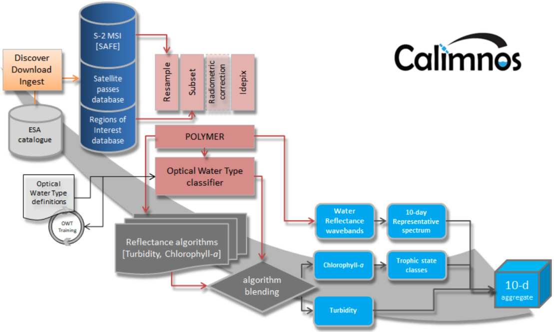
[Olmanson et al., 2016]



[Ocean Optics Web Book]

Standard processing chain

- Many steps and by-products from signal/image acquisition to the final product
- Transmission → Preprocessing → Processing
- A wide diversity of problems and dedicated tools



[Stelzer et al., 2020]

Challenges

- How to select the best features that describe the problem
- Extract the best combination of spectral bands
- Automatically find groups of pixels in the image for screening, detection...
- Estimate geo-physical variables from the spectra (e.g chlorophyll-a)
- Assign semantic classes to objects/regions in the scene (e.g. water types)
- High dimensional data: multi-temporal, multi-angular and multi-source fusion
- Non-linear and non-Gaussian feature relations
- Few supervised (labeled) information is available (high cost)
- Tons of data to process in (near) real-time

Feature Extraction

Feature selection

Try to find a subset of the input variables (also called features or attributes).
Extracting features from remote sensing images is essential to:

- Compress information for storage/transmission
- Reduce (spatial and spectral) redundancy
- Make image processing algorithms more robust (to noise, labels, dim.)
- Visualize data characteristics
- Understand the underlying physical relations

Extracted features can be either:

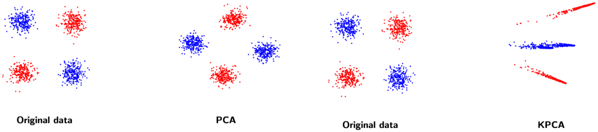
1. Spectral:
   - ~~Physically-based spectral features: erosion, dilation, filters~~
   - Statistical multivariate methods: linear and nonlinear
2. Spatial/contextual:
   - ~~Standard image processing descriptor~~

Spectral feature extraction

- Measured spectral signal at the sensor depends on the illumination, the atmosphere, and the surface observed
- Physically-inspired features before applying a machine learning algorithm
- Adapt standard feature extraction methods, such as PCA, to include knowledge about the physical problem

Statistical multivariate methods:

- Dimensionality reduction is sometimes essential before classification or regression
- Most of the spectral feature extractors are based on multivariate analysis: “project data onto a subspace that maximize explained variance, minimize correlation, minimize error, etc.”
- Linear (PCAs) and non-linear methods (KPCAs)



[PCA and kernel PCA explained]

Principal component analysis (PCA)

- Find features maximizing the variance of the data
- The Python code:

PCA transformation

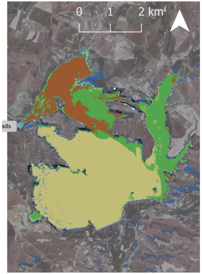
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
prinComp = pca.fit_transform(input_data)
prinDf = pd.DataFrame(data = prinComp,
            columns = ['PCA␣1', 'PCA␣2'])
```

- Advantages and disadvantages:
  - ✓ Simplicity
  - ✓ Easy to understand
  - × Unsuitable for non-linear problems (kPCAs?)
  - × More dimensions than points?

Classification

In a nutshell

How to pass from here...



...to here?

Challenging problem!

- What is a class? How many classes in the scene?
- What is "turbid" water?
- Do I have enough labeled data?
- Can my classification model be generalized?
- Labeled data is expensive
- Different atmospheric and illumination effects on several scenes
- Do I need atmospheric correction?
- Expert knowledge is needed pre- and postprocessing

Statistical classifiers: a summary

Parametric

Assume a particular density distribution
LDA, GMM

Non-parametric

No assumption about the data distribution
k-NN, NNETS, TREES, SVM

Supervised

Need labeled input-output pairs
LDA, k-NN, TREES, SVM

Unsupervised

No need labels
k-means, EM-GMM, SOM

Semisupervised

Use both labeled and unlabeled data
Laplacian SVM, TSVM, graphs

One-class

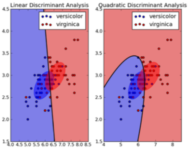
Interest in detecting just one class
SAM, OSP, RX, OC-SVM

- Not too much success in parametric classifiers, as some assumptions break
- Currently, nonparametric classifiers and committees of experts excel!
- k-NN: good compromise between accuracy and computational cost
- Support vector machines (SVM) typically outperform the rest

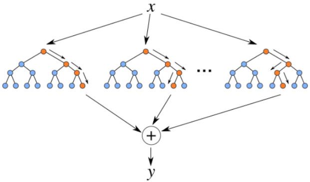
Most common supervised classifiers (for OC)

The easiest way to achieve interpretability is to use only a subset of algorithms that create interpretable models.

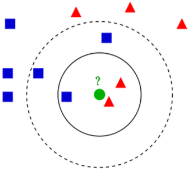
- Linear discriminant analysis (LDA)



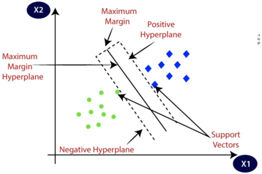
- Random Forest (RF)



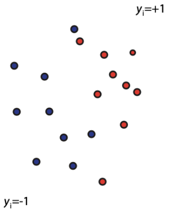
- k Nearest Neighbor (k-NN)



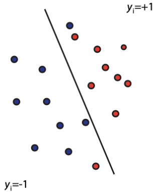
- Support Vector Machines (SVM)



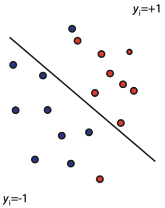
The linear case



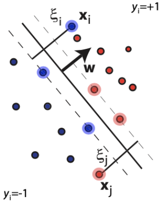
Input data



Another solution



One solution



Optimal case

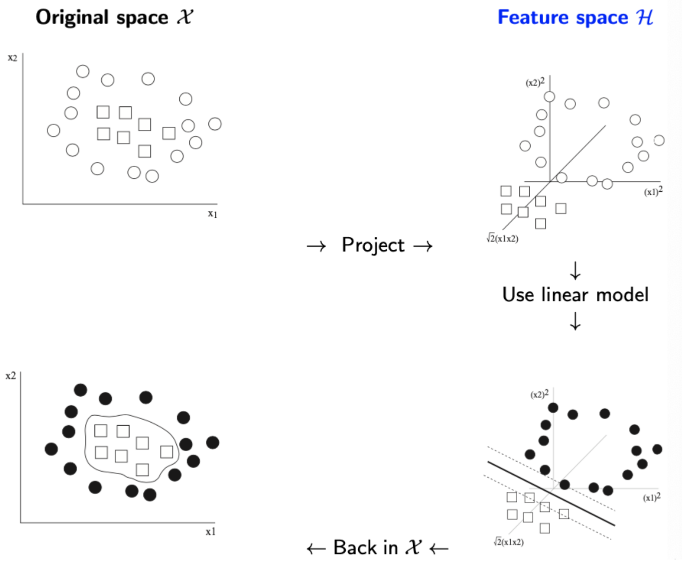
Example of LDA in Python

Linear Discriminat Analysis

```
from sklearn.datasets import make_classification
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative
    =10, n_redundant=0, random_state=1)
# define model
model = LinearDiscriminantAnalysis()
# fit model
model.fit(X, y)
# define new data
row = [0.12777556,-3.64400522,-2.23268854,-1.82114386,
1.75466361,0.1243966,1.03397657,2.35822076,
1.01001752,0.56768485]
# make a prediction
yhat = model.predict([row])
# summarize prediction
print('Predicted Class: %d' % yhat)
```

Machine Learning Mastering

Non-linear classifiers



Example of SVM in Python

Support Vector Machines (SVM): “non-parametric kernel method that fits an optimal linear hyperplane separating the classes in a higher dimensional representation (feature) space”

Perform binary classification using non-linear SVC with RBF kernel

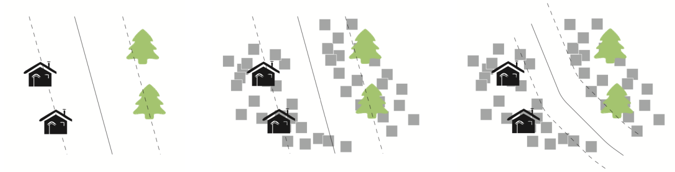
```python
from sklearn.datasets import make_classification
from sklearn import svm
xx, yy = np.meshgrid(np.linspace(-3, 3, 500),
                     np.linspace(-3, 3, 500))
np.random.seed(0)
X = np.random.randn(300, 2)
Y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0)

# fit the model
clf = svm.NuSVC(gamma='auto')
clf.fit(X, Y)
```

scikit-learn.org

Not enough labeled data?

- The image statistics can be misinterpreted because there are not enough representative pixels per class.
- We can include information from unlabeled samples in what is known as semi-supervised learning.
- For instance using Bagged kernels

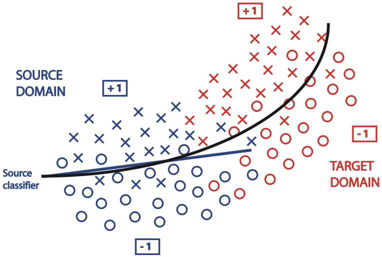


Supervised (SVM) Unlabeled distribution Semi-supervised classifier

Advanced RS classifiers

We would like to migrate classifiers across space and time and we do not have labeled information of all the images to be used. What can we do?

- Select invariant features
- Tune the classifier to the new domain
- Adapt data representation to match the input spaces (SS-MA)
- Select good samples in the new domain for sampling with active learning (AL)



[Matasci et al., JSTARS 2012]

Python exercise

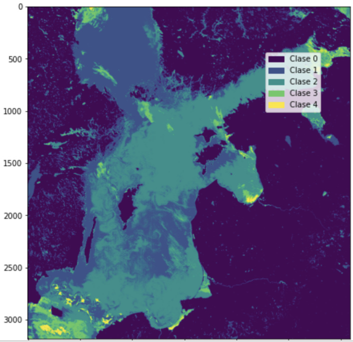
We will work on Google Colab

2nd_step_classify_image_S3_with_models.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 November

+ Code + Text

```
import matplotlib.patches as mpatches
values = np.unique(predictions.ravel())
plt.figure(figsize=(16,8))
im = plt.imshow(predictions,vmin=0,vmax=4)
colors = [im.cmap(im.norm(value)) for value in values]
patches = [mpatches.Patch(color=colors[i], label="Clase {l}".format(l=values[i]) ) for i in range(len(values)) ]
plt.legend(handles=patches, bbox_to_anchor=(1.05, 1), loc=0, borderaxespad=6)
plt.show()
```

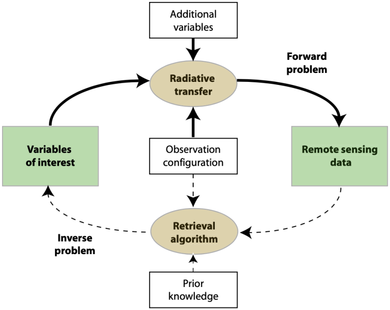


Regression

Retrieval of biophysical parameters

The objective here is to transform measurements into biophysical parameter estimates with EO data.

- Forward modelling: simulate a database of pairs of reflectance spectra + parameters with RTMs
- Inverse modeling: numerical/statistical invert models from RS data to estimate the parameters by designing algorithms that, starting from radiance, can give estimates of the variables of interest



Methods for model inversions

Three main families of methods:

1. *Statistical* inversion models: parametric and non-parametric
    - Parametric models rely on physical knowledge of the problem and build explicit parameterized expressions that relate a few spectral channels with the bio-geo-physical parameter(s) of interest.
    - Non-parametric models are adjusted to predict a variable of interest using a training dataset of input-output data pairs.
2. *Physical* inversion models: try to reverse RTMs
    - After generating input-output (parameter-radiance) datasets, the problem reduces to, given new spectra, searching for similar spectra in the dataset and assigning the most plausible ('closest') parameter.
3. *Hybrid* models try to combine the previous approaches.

Statistical approaches

Two main approaches:

1. **Parametric regression** inversion models: assume an explicit model for retrieval, e.g. discrete band approaches like indices, band ratios...
2. **Non-parametric regression**: do not assume explicit feature relations

Linear non-parametric models
Stepwise multiple linear regression (SMLR)
Partial least squares regression (PLSR)
Ridge regression (RR)
Least Absolute Shrinkage and Selection Operator (LASSO)

Non-linear non-parametric models
Decision trees, bagging and random forests
Neural networks
Kernel methods: SVR, RVM, KRR, GPR
Bayesian networks

Parametric approaches

Weaknesses

- Makes only poorly use of the available information within the spectral observation; at most a spectral subset is used. Therefore, they tend to be more noise-sensitive as compared to full-spectrum methods
- Parametric regression puts boundary conditions at level of chosen bands, formulations and regression function.
- Statistical function accounts for one variable at the time.
- A limited portability to different measurement conditions or sensor characteristics.
- No uncertainty estimates are provided. Hence the quality of the output maps remain unknown.

Strengths

- Simple and comprehensive regression models; little knowledge of user required.
- Computationally inexpensive.
- Fast in processing.

Non-parametric approaches

Weaknesses

- Training can be computational expensive.
- Can create over-complex models that do not generalize well (overfitting).
- Expert knowledge required, e.g. for tuning. However, toolboxes exist that automate some steps.
- Some regressors behave rather unstable when applied to data that deviate from statistically different from those used for training.
- Most of them act as a black box.

Strengths

- Can make use of all bands, full spectrum.
- Build advanced, adaptive (nonlinear) models.
- Some methods cope well with redundancy and noisy data.
- Once trained, fast processing images.
- Some of them (e.g. NN, decision trees) can be trained with high numbers of samples.
- Some methods provide insight in model development (e.g. GPR: relevant bands; decision trees: model structure).
- Some methods provide uncertainty intervals (e.g. GPR, KRR).

How to measure the goodness of a model?

Given two variables y_i and $\hat{y}_i$, i = 1, ..., N

- Error (residuals): $e_i = y_i - \hat{y}_i$
- Bias: mean error (ME): $ME = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$
- Accuracy: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
- Goodness-of-fit: Pearson‘s correlation coefficient

In Python:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
print("r2=",r2_score(ytest, ypred))
print("MAE =", mean_absolute_error(ytest, ypred))
print("RMSE =", mean_squared_error(ytest, ypred))
```

Advanced statistical retrieval

1. What is Gaussian Process (GP):
   - Gaussian Processes are a generalization of the Gaussian probability distribution.
   - It is a probability density over functions, non-linear and non-parametric.
   - GPR is still a form of supervised learning.
   - A Gaussian process is a Gaussian random function, and is fully specified by a mean function m(x) and covariance function k(x, x). This covariance function is called the latent function or the “nuisance” function.
   - They are a type of kernel model and they are capable of predicting highly calibrated class membership probabilities
   - The hyperparameters of the GPR algorithm on a given dataset can be tuned.
2. Deep Learning (it is all about scale)
   - Deep Learning is a subfield of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.
   - Multi-output regression involves predicting two or more numerical variables.
   - Deep learning neural networks are an example of an algorithm that natively supports multi-output regression problems.
   - Neural network models for multi-output regression tasks can be easily defined and evaluated using the Keras deep learning library.
   - What is deep learning?

Summary on regression

- Biophysical parameter estimation is perhaps the most important (and challenging) problem in remote sensing
- Traditional methods were focused on simplistic approaches using only few spectral bands
- New regression-based approaches alleviate the problems by exploiting the wealth of spectral and auxiliary information
- The common approaches consider:
  - Empirical models (e.g. indices) are easy, fast but too general
  - Physical radiative transfer models are flexible but slow and require specific information (e.g. aerosols, geometry) which is not always available
  - Non-parametric regression may offer a robust alternative that can be easily implemented in operative processing chains

Python exercise

We will work on Google Colab

Bbp retrieval using BGARGO + S3OLCI_v2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Notebook outline

- 1 - Load data & Overview
- 2 - Separate data in Input/Output
- 3 - Standarization
- 4 - Split in training and test
- 5 - Training the models
- 6 - Relevance of features in Random Forest Rgressor model
- 7 - Plot Bbp Profiles

Import libraries

```
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from sklearn.linear_model import Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

1. Load Sentinel-3 OLCI/BGC-Argo data & Overview

```
Dataset = pd.read_csv('Retrieval_Dataset.txt', sep = '\t')
```

```
# Allows to visualize all columns of the dataset (None = Active / False = Default visualization parameters)
pd.set_option("display.max_columns", None)
```

```
#Data Frame dimensions
Dataset.shape
```

```
(411, 83)
```

Thoughts

- A major disadvantage of using machine learning is that insights about the data and the task the machine solves is hidden in increasingly complex models.
- The higher the interpretability of a machine learning model, the easier it is for someone to comprehend why certain decisions or predictions have been made. Miller (2017), "Interpretability is the degree to which a human can understand the cause of a decision". Another one is: "Interpretability is the degree to which a human can consistently predict the model's result".

Links and references

Brockmann Consult GmbH

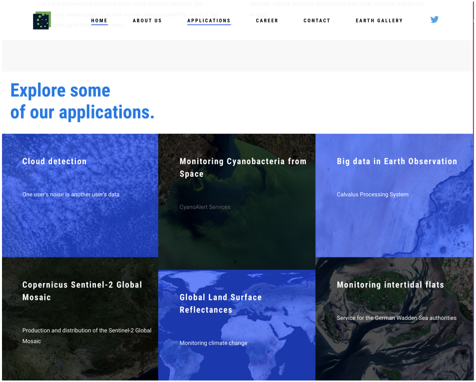
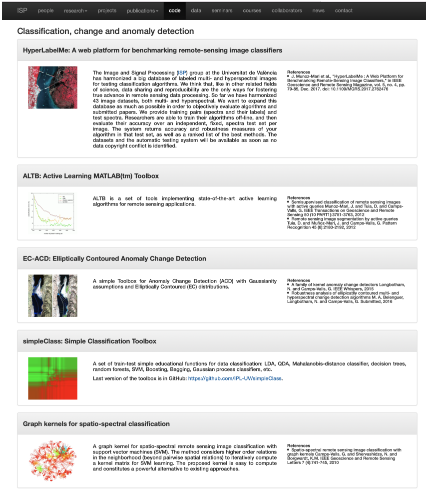


Image and Signal Processing Group, Universitat de València



[Special thanks to G. Camps-Valls and D. Tuia]