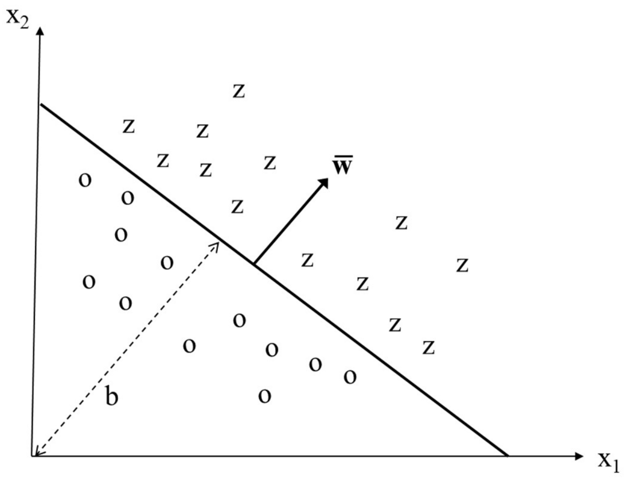
Lab Tutorial
Machine Learning for Beginners



This publication along with data set can be downloaded from this homepage,
http://apachepersonal.miun.se/~bentho/mlb/index.htm

1. Motivation

A real-world problem scenario is selected for this lab assignment, that way hopefully making the study of machine learning more interesting. Plastic packaging material must be sorted by its type of polymer before it can be recycled. A hyperspectral camera working with infrared light can be used for imaging of plastic pieces that passes by the camera in real time. Detection of type of polymer is accomplished by using machine learning on the spectral data recorded by a camera operating in short wave infrared region (SWIR). Enjoy this video showing a commercial sorting machine [1] .

2. About using Matlab for machine learning

Matlab is a general tool for computation and simulation. Matlab includes graphical user interfaces (GUI) for selected problem areas such as synthesis and implementation of filters or training and implementation of machine learning. Instead of using the GUI for machine learning, you should instead learn how to write scripts for importing data, provide plots for overview of data, perform training and classification. The benefit of using scripts is that the scripts provide good technical documentation that can be reviewed by your teacher. This is also the reason why you will be given hints in this tutorial on what kind of functions to use. General skills on Matlab modelling using vectors and matrices, plotting of data is best achieved by following a general Matlab tutorial [1] [3] . General Matlab modelling is not difficult to learn if you have had any previous courses in imperative programming such as C, C++, Python or Java.

The time scheduled in the lab together with your teacher is four hours for this lab. However, you need to be prepared to spend much more working hours to be able to finish all tasks.

3. Hyperspectral imaging of plastics for recycling

3.1. Introduction

Hyperspectral imaging means that hundreds of spectral channels are used to sense the intensity of light at narrow wavelength bands. You can compare with a much simpler color camera that uses three channels for the elementary colors: red, green and blue. Hyperspectral cameras can be equipped with an imaging detector that is sensible for light within the invisible infrared region. Using a range of e.g. 0.9 to 1.7 µm wavelength allow for identification of materials such as polymers. Light reflected in the surface of the analyzed material show unique spectral signatures caused by vibrational modes of the molecules. This kind of material analysis is called vibrational spectroscopy and is stemming from analytical chemistry where another type of instrumentation based on Fourier transform is used. Figure 1 depicts a principle sketch of a push-broom hyperspectral line camera. One line of pixels is imaged such that the sensing of photons in each pixel is divided into N spectral channels, λ_1 to λ_N .

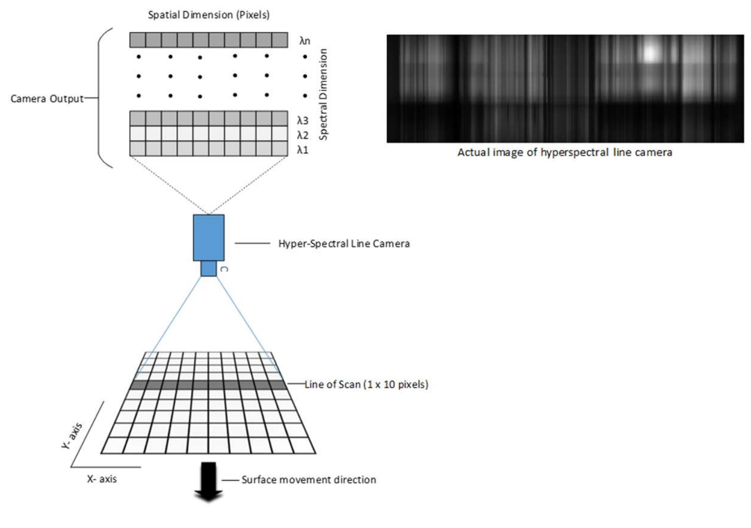

Spatial Dimension (Pixels)
λn
Spectral Dimension
Camera Output
λ3
λ2
λ1
Actual image of hyperspectral line camera
Hyper-Spectral Line Camera
Line of Scan (1 x 10 pixels)
Y- axis
X- axis
Surface movement direction

Figure 1. Push-broom hyperspectral line camera.

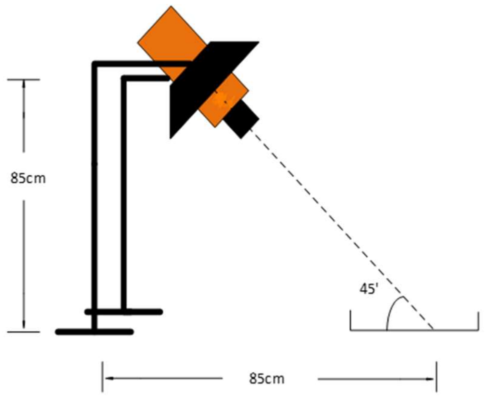
3.2. Experiment on polymers

The data set presented to you in this lab assignment is generated from using a hyperspectral camera in accordance with the principle experiment setup shown Figure 3. A halogen lamp is used to generate light within both (visible) and infrared region such that an area in front of the camera is illuminated. The hyperspectral camera is sensitive to 0.9 to 1.7 µm wavelengths. Samples of plastic pieces from packaging material is placed in front of the camera on the floor and within the single line that is imaged by the camera. As you will be able to see in your experiments, different plastics generate spectral profiles that are unique for its polymer. Relative reflectance is computed for all samples using a calibration target. This calibration is used to suppress the impact of variations of sensitivity in the camera and for variations in intensity distribution of illumination [5] .

Plastic packaging material is required to have a label on it for indication of what kind of polymer it consists of. Have a look at the example of a triangular mark and the list of codes shown in Figure 2. The same numbers as in this list are also used as class labels for the provided data set.

(1) PETE
(2) HDPE
(3) PVC
(4) LDPE
(5) PP
(6) PS
(7) Background

Figure 2. Recycling codes 1-6. Code 7 is used to label the background (floor) at imaging.



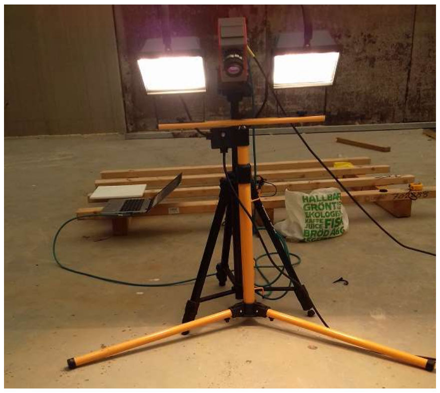


Figure 3. Experiment setup.

4. Overview of input data

The purpose of this first experiment is to get an overview of the dataset generated from the polymer experiment previously described.

4.1. Read data and generate plots of spectral signatures

The first command that always should be used in a Matlab script is to clear the workspace from stored data. All active figures should also be deleted. Data vectors, labels and a wavelength scale can then be imported into the workspace using the *load* command. The data file *dataVectors.mat* is provided for you [4] . Make a single plot that shows the mean relative reflectance for each class of polymers, 1 to 6, also include background 7. This plot will give you an indication of how unique the spectral profiles are. Use unique line colors for each plotted polymer. Use the vector *scale* to display the correct wavelength on the horizontal axis.

Here is a list of useful Matlab commands that you can explore for this task:

- size
- length
- class
- clear all
- close all
- clc
- load
- plot
- hold on
- xlabel
- ylabel
- legend
- title
- find
- mean

Remember that you can always find information about a specific command by typing, *help command_name* in the Matlab command window.

4.2. Perform PCA and make a perspective 3D plot of score data

You should now perform principal component analysis on the provided data set. This is an example of how to use the *pca* command,

```
[coeff, score, latent, tsqared, exp] = pca(vectors, 'Centered', 'on', 'VariableWeights', 'variance', 'NumComponents', noComp);
```

The arguments *Centered =on* and *VariableWeights=variance* enables centering and scaling as pre-processing. Use command plot3 for the perspective 3D plot. The following arguments given to *plot3* generates a small dot for each score vector, *'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',4*. The letter 'b' means blue color. Use unique colors for each polymer class.

4.3. Make a 2D plot of loading coefficients versus wavelength

The *pca* command returns a matrix for the coefficients, a column of coefficient values for each principal component. The number of rows for this matrix should be the same as the length of vector *scale*. Make a plot for the first four PC, coefficient loadings versus wavelength, use vector named *scale*. It is possible to analyze from this graph, which wavelengths are the most important for computing PC1 to PC4. How much of the data variance is explained by the first four PC?

5. Classification using Support Vector Machine

This task is devoted to the training of a multi-class svm-classifier and the classification of spectral signatures for the six polymers. A multi-class svm is actually built from several binary svm classifiers combined with an additional algorithm that generates labels for the classifier output. You should use approximately 80% of provided data vectors for training and the remaining 20% for testing. The script that you are going to write is preferably divided into four consecutive parts:

- Read and shuffle
- Divide vectors
- Train and classify
- Validate

The following four subsections will give you more detailed hints on how to proceed:

5.1. Read and randomly shuffle all data vectors

Begin this new Matlab-script in the same way as in section 4.1 by loading provided data. Get the dimensional size of the matrix holding all data vectors, *[m,n] = size(vectors)*;
The number of vectors is *m* and the length of all vectors is *n*. Next step will be to generate random integer values between 1 and *m*, *idx = randperm(m);*
The vector *idx* of length *m* can now be used to randomly reorganize both data vectors and labels,

```
vectors(:,:) = vectors(idx,:);
labels = labels(idx);
```

Remember that the key information about which class a certain vector belongs to is still explained by the vector *labels* of length *m*.

5.2. Divide all data vectors into separate sets of training- and test vectors

Introduce a parameter in the beginning of your script such that you can easily change the fraction of training vectors, start with approximately 80% training vectors, *trainSet* and *testSet*. Do not forget to equally divide the vector of labels into vectors *testLabels* and *trainLabels*.

5.3. Train a multi-class svm and perform classification

Start by specifying the svm to be used, *T = templateSVM('Standardize',1,'KernelFunction','rbf');*
The selected kernel function is *radial basis function* (rbf). Preprocessing by centering and scaling of whole data set is enabled. Next step will be to train the classifier, *Mdl = fitcecoc(trainSet,trainLabels,'Learners',T,'Verbose',2);*
The data structure *Mdl* holds the trained classifier and can be used for later classification. The argument *Verbose* is set to generate tracing information during the learning process.
Perform classification of the *testSet* using the multi-class svm classifier described by Mdl, *cl = Mdl.predict(testSet);*
The generated vector *cl* contains the labels of classified output. The next step will be to analyze the classified labels with the know true labels.

5.4. Validate the classified output

An example of how to find out how many vectors out of *testSet* that were erroneously classified is, *err = sum(cl ~= testLabels);* You should calculate the overall percentage of success rate using this error count.

However, the success rate is not giving any information about possible problematic classes, frequently mistaken for any of the other classes. Therefore, we should analyze the classifier output for each input class separately. Firstly find out indexes to the vectors corresponding to a single input class. In this example to class 1, *ind1 = find(testLabels == 1);* All classifications for input class 1 can now be selected using *cl(ind1)*. We can expect this selection to contain mistakes, hence labels other than 1. These mistakes can be analyzed by a histogram function, *[result,EDGES] = histcounts(cl(ind1), [1 2 3 4 5 6 7 8]);* Counts for seven integer bins will be output in the vector *result*. If the classification was without errors (for this example), all counts will be in the first bin. Repeat this analysis for all seven input classes and you will have all the data needed to prepare a confusion matrix. Table 1 shows an example on how you can present a confusion matrix including the overall success rate.

Table 1. Example of a confusion matrix

True Input	Classified Output				Sum of all
	Class 1	**Class 2**	**Class 3**	**Class 4**	
Class 1	**50529**	0	7015	672	58216
Class 2	0	**58278**	0	0	58278
Class 3	4842	0	**53220**	471	58533
Class 4	4939	0	1015	**52567**	58521
Overall success rate	92 %				

6. Classification using K-Nearest Neighbors

You should now repeat the same experiments as in previous section 5 on svm classification. Copy the script from section 5 and just replace classifier initialization and training at 5.3 with a knn classifier,
Mdl = fitcknn(trainSet,trainLabels,'NumNeighbors',5,'Standardize',1);
You can try different numbers of neighbors to include into the search, which most likely will affect the overall success rate.

7. Simulate optical band pass filters

A question that you should ask yourself; is it possible to use a more simple detector instead of a hyperspectral camera and still achieve a reasonable success rate? Can hundreds of the camera's spectral channels be reduces in to three or maybe four spectral bands? Study the graphs that you generated in sections 4.1 and 4.3 and try to identify those three to four bands that are the most important for the PCA model. Compute mean values of the spectral channels within those selected wavelength bands and use the new data vectors is input for the same svm classification as in section 5. Compare with previous results.

References

[1] Polymer sorting machine, https://youtu.be/mLya2NuY4Yk
[2] Matlab learning tutorial, http://www.cyclismo.org/tutorial/matlab/
[3] Matlab learning tutorial, http://users.rowan.edu/~shreek/networks1/matlabintro.html
[4] Data set, http://apachepersonal.miun.se/~bentho/mlb/download/dataVectors.mat
[5] M.S. Shaikh et.al, "Calibration of a Hyper-Spectral Imaging System Using a Low-Cost Reference", https://www.mdpi.com/1424-8220/21/11/3738