UNIVERSITAT DE BARCELONA

Data Analysis and Multivariate Regression Modelling with Python for Gas Concentration Prediction: An Array of Commercial MOX Sensors as an Example of Chemical Sensing Unit

OBJECTIVES

Study the impact of humidity in Multivariate Prediction Models trained with data coming from an array of MOX sensors. Base

MATERIAL

- Python 3.7 or higher
- Packages:
    - Sys
    - Numpy
    - Pandas
    - Matplotlib
    - Sklearn
    - Scipy
- Spyder (optional but recommended)

Dataset

We will use a public dataset for this lab. The dataset contains recordings from 14 MOX sensors exposed to gas mixtures of carbon monoxide and humidity in a small sensor chamber (Fig. 1). Sensors #1-7 are from FIGARO manufacturer (model: TGS 3870-A04) while sensors #8-14 are from FIS manufacturer (model FIS SB-500-12).

Figure 1. Gas sensor array and gas chamber

The dataset is a csv file with the following format:

```
Time (s),CO (ppm),Humidity (%r.h.),Temperature (C),Flow rate (mL/min),Heater voltage (V),R1 (MOhm),R2 (MOhm),R3 (MOhm),
0.0000,0.0000,49.7534,23.7184,233.2737,0.8993,0.2231,0.6365,1.1493,0.8483,1.2534,1.4449,1.9906,1.3303,1.4480,1.9148,3.4
0.3090,0.0000,55.8400,26.6200,241.6323,0.2112,2.1314,5.3552,9.7569,6.3188,9.4472,10.5769,13.6317,21.9829,16.1902,24.278
0.6180,0.0000,55.8400,26.6200,241.3888,0.2070,10.5318,22.5612,37.2635,17.7848,33.0704,36.3160,42.5746,49.7495,31.7533,5
0.9260,0.0000,55.8400,26.6200,241.1461,0.2042,29.5749,49.5111,65.6318,26.1447,58.3847,67.5130,68.0064,59.2824,36.7821,6
1.2340,0.0000,55.8400,26.6200,240.9121,0.2030,49.5111,67.0368,77.8317,27.9625,71.7732,79.9474,79.8631,62.5385,39.6271,6
1.5440,0.0000,55.8400,26.6200,240.8361,0.2020,60.1083,74.3444,81.5100,29.7970,72.9643,83.1477,80.5302,58.0412,39.2482,6
1.8540,0.0000,55.8400,26.6200,240.7602,0.2010,64.1020,74.3444,76.4748,28.1797,72.4181,78.4368,79.0768,59.7614,40.4067,6
2.1630,0.0000,55.8400,26.6200,240.6845,0.2009,62.6869,71.3877,73.8965,27.6523,64.4007,69.7912,72.5239,55.6363,39.6271,6
```

Each row is a sample (Fs=3 Hz) and contains the time of acquisition, the gas concentration, humidity level and temperature inside the chamber, flow rate, heater voltage and the resistance of the 14 sensors.

During this experiment, the sensors were exposed to 100 gas mixtures with gas concentrationsin the range 0-20 ppm and relative humidity in the range 20-70 % r.h. (Fig. 2)

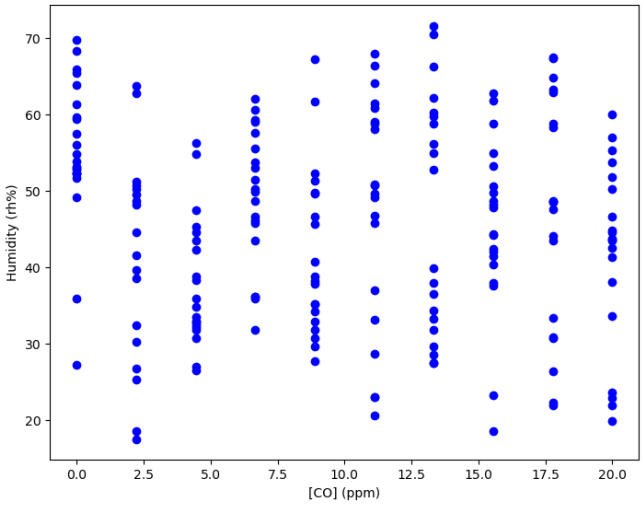

Figure 2. Experimentalconditions (N=200)

Due to the dynamics of the gas chamber, the gas mixtures need around 15 minutes to stabilize (Fig. 3). Therefore, each condition is maintained for 15 minutes.

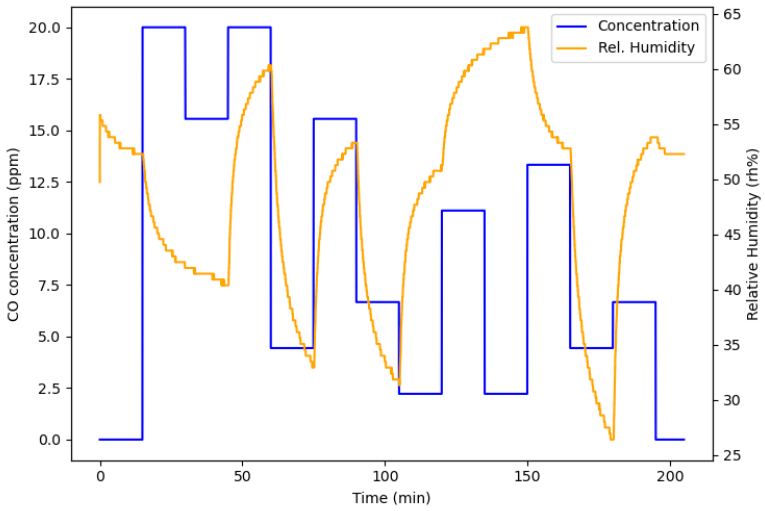

Figure 3. Evolution of the gas mixtures inside the chamber during the first 3 hours of experiment.

The sensors were operated in temperature modulation, with a squared heating waveform with voltages between 0.2 and 0.9 V in periods of 20 and 25 s. (Fig. 4)

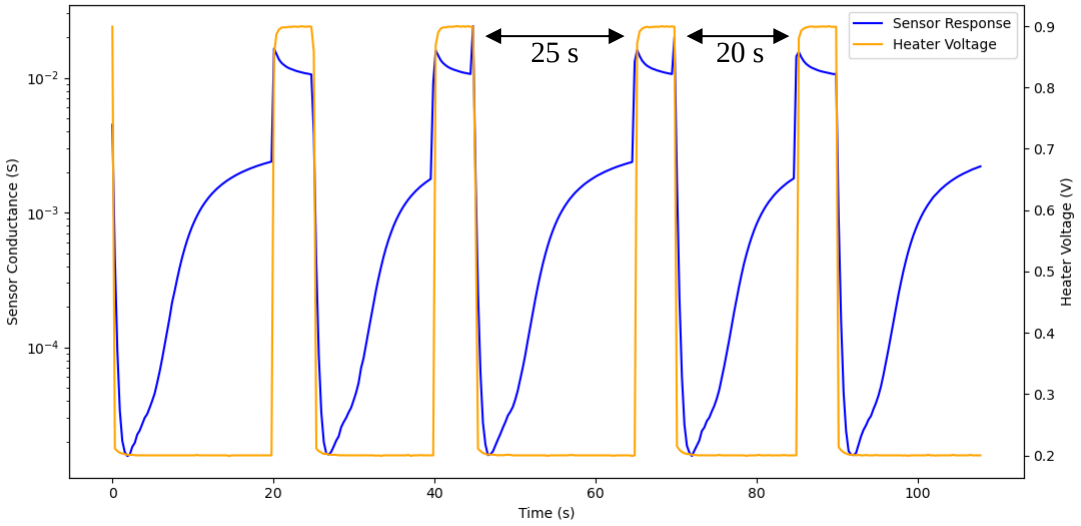

Figure 4. Conductance of sensor #13 and heater voltage during the first 100 s of measurement

In each condition, there are approximately 40 heating cycles (Fig. 4). The raw signals contain some noise (spikes).

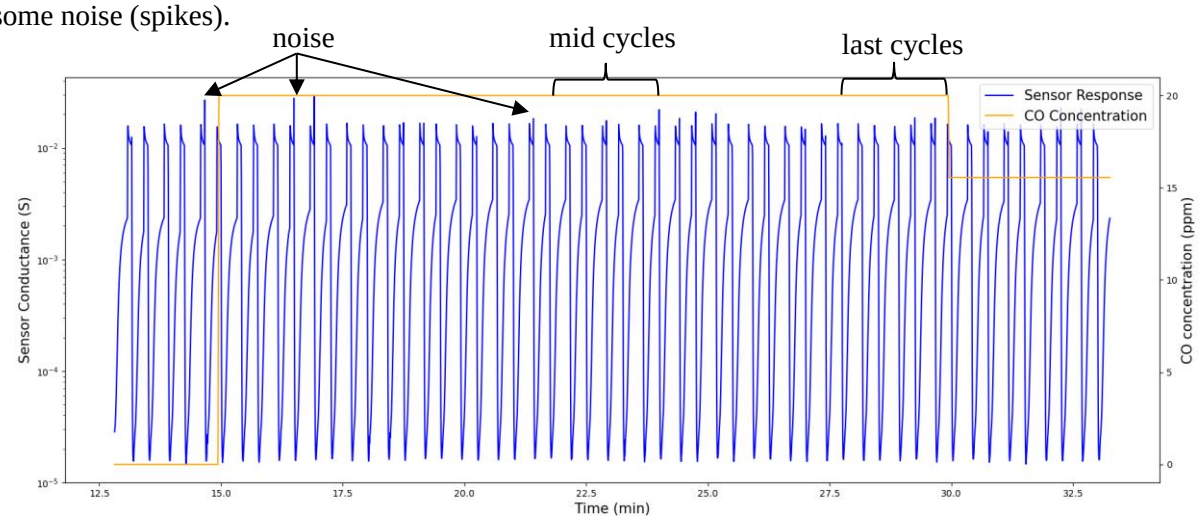

Figure 5. Sensor response during two full conditions. There are 40 cycles in each condition.

Development

The objectives of the practical session are:

1. Preprocessing: students will remove the noise of the signals, segment all heating cycles from the raw continuous signal and average the last 3 and the central 3 cycles of each condition.
2. Train a PLS model at constant humidity: students will peak up a representative humidity and train a PLS model with 10-fold Cross-Validation.
3. Study the performance at different humidity: students will test the performance of the trained model when the concentration is predicted from data acquired at different humidity.
4. Train a more general model: students will train a new model accordingly with the workflow presented in Fig. 6. The Blind Samples will be the 30% of the complete dataset.

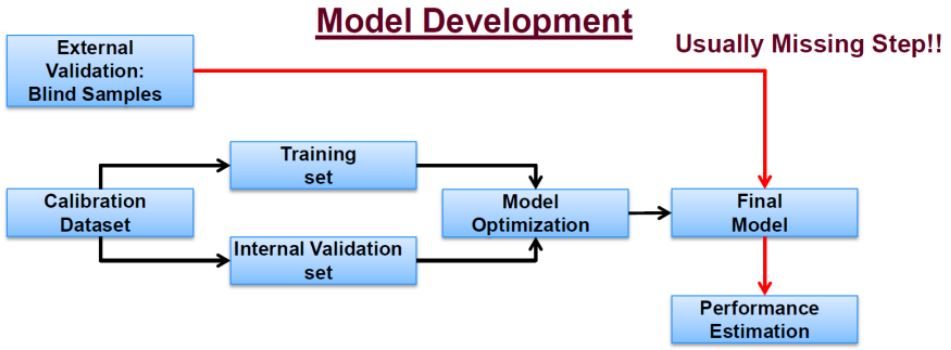

Figure 6. Multivariate Prediction Model Training Workflow.

Preprocessing

1) First, we will read the dataset and assign shorter names to variables.

```
import pandas as pd
data = pd.read_csv(url)
  # Get features
dataTop = data.columns
t = data[dataTop[0]].values
rh = data[dataTop[2]].values
hv = data[dataTop[5]].values
co = data[dataTop[1]].values
moxCol = dataTop[6:]
R = 1/((10**3)*data[moxCol])
```

2) To remove the noise, we will use a combination of median filtering and low-pass filtering. You might use rolling().mean and rolling.median() for this purpose.

3) To segment the heating cycles from the raw signals, you might use the derivative to identify the rising edges of the heater voltage. Also, to identify the segmentation point you might use the find_peaks in scipy

```
from scipy.signal import find_peaks
```

4) To extract the portions of the sensor response between every two consecutive peaks you might, for every sensor, go through the total response, selecting the set of data of each portion. In order to end up with the same number of samples per portion you might perform an interpolation. Here you have a raw version:

```
for i in range(0,len(peaksHV)-1,1):
      tx = t[peaksHV[i]:peaksHV[i+1]-1]
      x = filtMox[:][peaksHV[i]:peaksHV[i+1]-1]
      tint = np.linspace(tx[0],tx[-1],Nsamp)
      oneIntR = pd.DataFrame()
      for col in moxCol:
          xd = x[col].values
          xI = np.interp(tint,tx,xd)
          df = pd.DataFrame({col:xI})
          oneIntR = pd.concat([oneIntR,df],axis=1)
```

UNIVERSITAT DE BARCELONA

5) To average 3 portions at the middle and at the end of the condition, you might have to go through every sensor response and select from the corresponding 40 cycles those your interested in. Consider that, during the condition, in 3 cycles, humidity might show changes. Here you have a raw version of the code:

```
for col in moxCol:
     a=X[col].values
     b=pd.DataFrame(a.reshape((int((len(X))/Nsamp)),Nsamp))
     oneMox=pd.DataFrame()
     for j in range(1,int(np.floor(2*len(b)/hvCycles))):
         aux3 = b.iloc[j*20-4:j*20-1].mean(axis=0)
         oneMox = pd.concat((oneMox,aux3.T),axis=1)
```

Partial Least Squares (PLS) at Constant Humidity

PLS regression is a multivariate regression method that bears some relation to principal components regression (PCR); instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space. Because both the X and Y data are projected to new spaces, the PLS family of methods are known as bilinear factor models.

PLS is used to find the fundamental relations between two matrices (**X** and **y**), i.e. a latent variable approach to modeling the covariance structures in these two spaces. A PLS model will try to find the multidimensional direction in the **X** space that explains the maximum multidimensional variance direction in the **y** space. PLS regression is particularly suited when the matrix of predictors has more variables than observations, and when there is multicollinearity among X values. By contrast, standard regression will fail in these cases (unless it is regularized).

The general underlying model of multivariate PLS is

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E}$$
$$\mathbf{y} = \mathbf{U}\mathbf{Q}^{\mathrm{T}} + \mathbf{F}$$

where **X** is an $n \; x \; m$ matrix of predictors, $\boldsymbol{y}$ is an $n \; x \; 1$ matrix of responses; **T** and **U** are $n \; x \; l$ matrices that are, respectively, projections of **X** (the **X** score, component or factor matrix) and projections of $\boldsymbol{y}$ (the $\boldsymbol{y}$ scores); **P** and **Q** are, respectively, $m \; x \; l$ matrices and $1 \; x \; l$ orthogonal loading matrices; and matrices **E** and **F** are the error terms, assumed to be independent and identically distributed random normal variables. The decompositions of **X** and **Y** maximize the covariance between **T** and **U**.

In order to obtain the final feature matrix, you might follow the steps presented below:

1) Preparing the data: as you would like to observe the performance of PLS prediction model at constant humidity, you might peak up the humidity with higher load of information (maximum number of CO concentrations). Fig. 8 shows the CO concentration vs relative humidity after the preprocessing performed in the previous section.

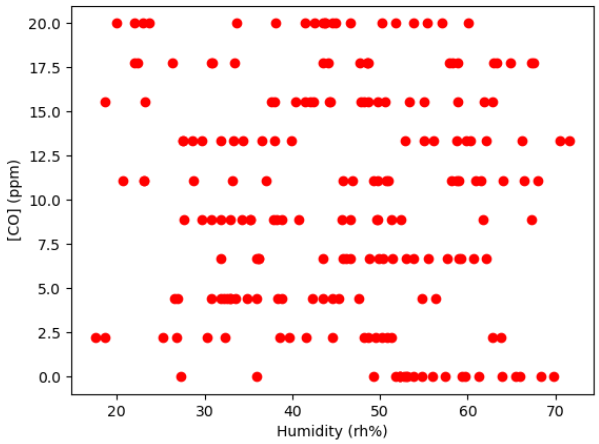

Figure 7. [CO] versus Relative Humidity after Preprocessing.

A raw example of the code is presented below:

```
rh1 = RHavg[0:2*Nsamp]
idxj = np.where((np.array(rh1)<=RHmax)&(np.array(rh1)>=RHmin))
```

2) Log transformation: you might want to include a logarithmic transformation before scaling in order to enhance little variations of the sensors response. Remember that sklearn package provides functionalities for that:

```
from sklearn.preprocessing import FunctionTransformer
def log_transform(x):
        print(x)
        return np.log(x)

transformer = FunctionTransformer(log_transform)
transformer.transform(X)
```

3) Mean centering: PLS assumes that **X** and **y** are mean-centered. Therefore, you might use the advantages of sklearn package to perform this centering. A raw example of how to use the sklearn scaler is presented below:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
scaler.transform(X)
```

Note that the standard scaler performs a mean-centering of data and a scaling of data using their standard deviation. You might deactivate these options by using `with_mean=False` or `with_std=False`.

UNIVERSITAT DE BARCELONA

4) 10-fold CV PLS training: a simple training of the model might be performed with the following code:

```
def simple_pls_cv(X, y, n_comp):
    #Run PLS with n_comp components
    pls = PLSRegression(n_components=n_comp)
    pls.fit(X, y)
    y_c = pls.predict(X)
    #Cross-validation
    y_cv = cross_val_predict(pls, X, y, cv = 10)
```

Note that the number of components of the PLS model is arbitrary. Remember that, sklearn package provides tools for calculating the R^2, RMSE and Explained Variance of a prediction.

EXERCISE 1

Generate a model with the number of components optimized for the given data at constant humidity. Get R^2, RMSE and Explained Variance scores for both, the training prediction and the Cross-Validation prediction and plot the evolution of these scores with the number of components. Once optimized, plot the prediction after calibration and the linear regression obtained after internal validation with the 10-fold CV.

EXERCISE 2

Validate your optimized model against the CO concentrations coming from other humidity and check the scores mentioned above against the scores obtained after inner validation.

Partial Least Squares (PLS) with Blind Validation

It is time to use the complete dataset obtained after the preprocessing. As we are increasing the amount of data, you might adopt the workflow presented in Fig. 6. You might want to split the dataset into train-set and test-set. Fortunately, sklearn package provides tools for doing that. A raw code for doing this is presented below:

```
from sklearn.model_selection import train_test_split
X_train, X_test,y_train, y_test = train_test_split(X,y,
                                  random_state=42,
                                  test_size=ts)
```

Remember, before training the model to perform the transformations you might consider useful.

EXERCISE 3

Generate a model with the number of components optimized for the given data for each type of sensors. Separate the complete dataset in train-set and test-set. Repeat EXERCISE 1 for the training set of each type of sensor and compare results. Get the relevant scores for a prediction over the test set and compare results with the obtained for the 10-folded CV inner validation. Finally, plot the linear regression for the test-set and compare with the actual concentrations for that set.