Synthetic Spectra Generation

Introduction

X-ray Photoelectron Spectroscopy (XPS) is a powerful analytical technique used to probe the chemical composition and electronic states of materials. It starts from passing photons towards the provided film and the electrons emitted out is captured via the energy analyser where the attracting force captures the fast approaching electrons and the repulsive force captures the slow electrons. This will be furtherly processed by the computer and deported out as a file formatted in .txt or .csv.

Why do we need to generate Artificial Spectra ?

Generating artificial spectra is necessary because real XPS datasets are limited, while machine learning requires large, diverse training sets. The simulated spectra enable training robust models for automated XPS analysis without the need for extensive experimental data.

So, Now you can find below the various steps in generating artificial spectra similar to the experimental data

1. **Peak Modeling**: Peaks are modeled to represent the intensity of emitted photoelectrons at specific binding energies corresponding to different elements or chemical states present in the sample. These peaks are typically represented by mathematical functions, such as Gaussian or pseudo-Voigt functions, which describe the shape and characteristics of the peaks.
2. **Parameters of Peak Functions**: The parameters of the peak functions, includes peak position, peak width (related to the uncertainty in energy), peak amplitude and number of peaks. These parameters can be varied to simulate different spectral features and sample compositions.
3. **Combining Peaks**: Once the parameters are defined, the individual peaks are combined to generate the overall spectrum. The number of peaks and their relative positions and intensities can be adjusted to simulate the complexity and diversity of real-world samples.
4. **Noise Addition**: Real XPS spectra are subject to noise and uncertainties arising from experimental conditions and instrumentation. To replicate this, random noise is added to the simulated spectrum. This noise can be generated using statistical distributions, such as the normal distribution, with parameters that control the noise level. In this case Pseudo Voigt function.
5. **Plotting**: The generated spectrum is then plotted, typically as a function of binding energy (or sometimes kinetic energy) along the X-axis and counts of electrons along the Y-axis. Visualization libraries like matplotlib, Plotly in Python are commonly used for creating plots of artificial spectra.

6. **Analysis and Interpretation**: Finally, the generated spectra can be analyzed using various data science techniques like ML and Modelling to extract meaningful information about the sample's composition, chemical states, and electronic properties.

Review: Artificial Spectra Generation from Iterative Peak Fitting Article

1. Import libraries
2. Inputs
3. Static variable for energy range -> x-axis
    a. Definition of peaks sizes of the sub databases, Specifically, there are sub-databases for 5, 4, 3, 2, and 1 peaksets, each with a different number of data points.
4. Function Definition for Artificial Spectra - Pseudo Voigt
5. To extract 5 peaks databases with 4 parameters of width, position, amplitude and no.of peaks
    a. np.random.rand() used for creating different size for each peaks
    b. Make a copy and convert to list
    c. Iteration of the peaks for deleting the entries that align closer which is factored by area
        i. To be noted, if two peaks are closer to each other the algorithm completely deletes the whole spectra which is checked corresponding to the matrix size.
    d. Storing the 5 peaks figure into the databases with added noise
    e. Finally, plotting the peaks across energy vs intensity
6. **This is the same procedure followed for all the peaksets from 1 to 4 and 5 (above)**
7. Assembling the sub databases into one for ML
8. Whole Database is saved in pickle and shuffled randomly for better results
9. Finally diving into ML - Split, Reshape and Store
    a. Splitting the Data
        i. Data split into three subsets, training, validation and testing
        ii. The original data consists of peaks represented as 2D arrays, where each row corresponds to a single peak and each column represents a point in the energy range.
        iii. The `train_peak`, `val_peak`, and `test_peak` variables are assigned portions of the original data based on predefined ratios (in this case, an 8:1:1 split). This means that 80% of the data is allocated to training, 10% to validation, and 10% to testing.
        iv. The purpose of splitting the data is to train the model on one subset (training set), tune hyperparameters on another subset (validation set), and evaluate the final model performance on the remaining subset (test set).
        v. The purpose of splitting the data is to train the model on one subset (training set), tune hyperparameters on another subset (validation set), and evaluate the final model performance on the remaining subset (test set).

b. Reshaping the CNN Input
    i. Before feeding the data into a convolutional neural network (CNN), it needs to be reshaped to fit the expected input format of the model.
    ii. CNNs typically expect input data to have a specific shape, which includes dimensions for the number of samples, height, width, and channels (for grayscale or RGB images).
    iii. In this case, the original 2D arrays representing the peaks are reshaped into 3D arrays by adding an extra dimension for the channel.
    iv. The reshaping converts each 2D array (representing a single peak) into a 3D array, where the additional dimension represents the channel or feature map.
    v. By reshaping the data in this way, it becomes compatible with the input requirements of CNN layers in the neural network model.

c. Variable Reassignment
    i. After reshaping each subset for CNN input, the reshaped arrays are stored back into variables with the same names (`train_peak`, `val_peak`, `test_peak`) to replace the original 2D arrays.
    ii. This reassignment ensures that the variables continue to hold the preprocessed data in the appropriate format for subsequent processing steps, such as model training, validation, and testing.

Critics

As the code is quite long with 800 lines, we can use some simple functions like np.random.uniform for better vision in the deletion of spectra when area is closer.

Other Exploration

1. Extension with various analysis and modeling softwares like
    a. BriXias Code
    b. CASA
    c. A-Analyser
    d. XPS Peak 4.1
    e. Thermo Avantage
2. Generated one synthetic spectra -> multiple spectra -> Addition of parameters like amplitude (under progress)etc. (link for a view)
3. Some notes in PhD day and Intern Seminar