sierras

Release 0.2.5

Francisco Fernandez

Jul 18, 2023

CONTENTS:

sierras is a tool for empirical Arrhenius equation fitting for thermally-induced physicochemical processes.

CHAPTER

ONE

REQUIREMENTS

You need Python 3.9+ to run sierras.

Authors
Francisco Fernandez (E-mail: ffernandev@gmail.com)

CHAPTER

TWO

REPOSITORY

https://github.com/fernandezfran/sierras/

CHAPTER

THREE

LICENSE

sierras is under MIT License

3.1 Installation

You can install the most recent stable release of sierras with pip

```
python -m pip install -U pip
python -m pip install -U sierras
```

3.2 sierras tutorial

In this tutorial we will use the **sierras** module to analyze the diffusion coefficient behavior as a function on temperature with the Arrhenius equation, we will extrapolate its value at room temperature and make a Arrhenius plot. All this in a straightforward way.

First we import the libraries needed

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

We use the experimental data of diffusion coefficients of lithium in silicon obtained by Fuller and Ditzenberger

[2]:
```
temperatures = np.array([1250, 1153.36, 1063.13, 970.65, 861.04, 769.34]).reshape(-1, 1)
diffusion_coeffs = np.array([7.72104e-6, 4.386714e-6, 2.23884e-6, 5.58574e-7, 5.15115e-7,
↪ 7.58213e-8])
```

We import the `ArrheniusRegressor` class from **sierras**

[3]:
```
from sierras import ArrheniusRegressor
```

As can be consulted in the module documentation, this class works in a `scikit-learn`-like way. Then, we can obtain the activation energy as an attribute of the object, plot agains the experimental data and get the results in a `pandas.DataFrame`.

We start by creating an arrhenius regressor object from the class with the desired constant value in the exponential of the Arrhenius equation (here the Boltzmann constant, as in the default case)

[4]:
```
k_boltzmann = 8.617333262e-5
areg = ArrheniusRegressor(constant=k_boltzmann)
```

3.2.1 fit the data

[5]:
```
areg.fit(temperatures, diffusion_coeffs)
```

[5]:
```
ArrheniusRegressor()
```

We can access to the activation energy (in this case k_B is in eV/K, so this is in eV)

[6]:
```
areg.activation_energy_
```

[6]:
```
0.7617837303407903
```

3.2.2 extrapolate to room temperature

We could use the predict method to extrapolate to any temperature, but room themperature is really relevant because the time required to obtain this value from a experiment or simulation may be prohibitive. This information is already calculated in `extrapolated_process_` attribute.

[7]:
```
areg.extrapolated_process_
```

[7]:
```
1.3856640816386593e-15
```

This result for the room diffusion coefficient is in the same units as the target data (in this case centimeters squared over seconds).

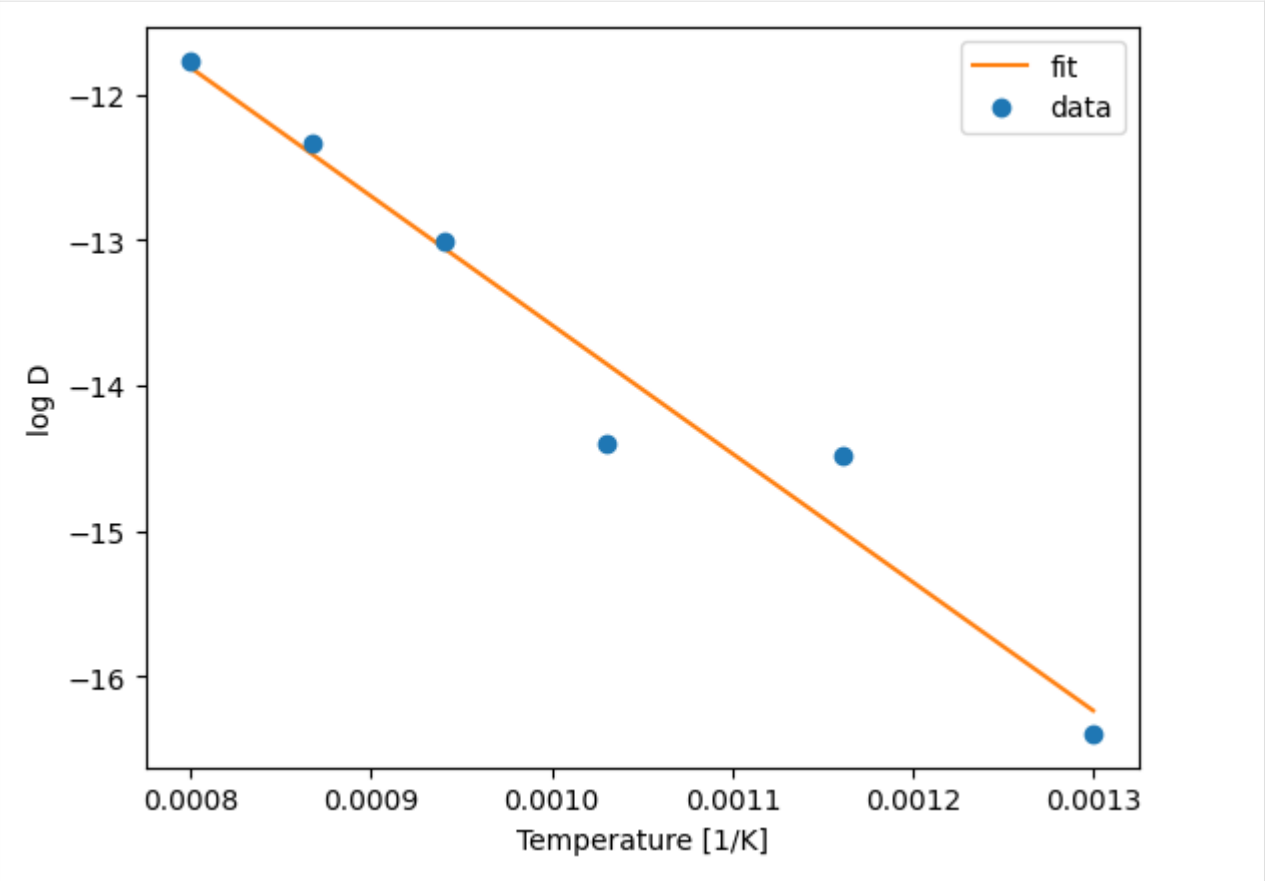
3.2.3 plot

[8]:
```
fig, ax = plt.subplots()

areg.plot(ax=ax)

ax.set_xlabel("Temperature [1/K]")
ax.set_ylabel("log D")
ax.legend()

plt.show()
```



3.3 sierras module

A tool for empirical Arrhenius equation fitting.

In physical chemistry, the Arrhenius equation is an empirical formula for temperature dependence of a process given by the following equation

$$k = k_0 e^{-E/k_B T}$$

where

- k is the thermally-induced process (diffusion coefficient, frequency of collision, crystal vacancies, among others),
- k_0 is the pre-exponential factor,
- E is the activation energy of the process,
- k_B is the Boltzmann constant, it could also be the universal gas constant R,
- T is the Temperature in Kelvin,

The exponential factor in this equation gives the probability that the process occurs and it denotes the fraction of atoms with energy greater than or equal than E.

Taking the natural logarithm of this equation yields to a linear relationship

$$\ln k = \ln k_0 - \frac{E}{k_B}\left(\frac{1}{T}\right)$$

that can be fitted and used to extrapolate the process to room temperature, which is usually difficult to obtain directly, or to get the activation energy from the slope.

For more details read: https://en.wikipedia.org/wiki/Arrhenius_equation and/or https://en.wikipedia.org/wiki/Arrhenius_plot

class sierras.**ArrheniusRegressor**(*constant=8.617333262e-05*)

Bases: BaseEstimator, RegressorMixin

Arrhenius equation fitting.

Parameters

constant (*float, default=8.617333262e-5*) – Either the universal gas constant, R, or the Boltzmann constant, k_B, default value is the later in eV/K units.

activation_energy_

Activation energy of the process, this is the same as `-self.constant * self.reg_.coef_[0]`.

Type

float

extrapolated_process_

The extrapolation at room temperature of the thermally-induced process, note that this is the same that `self.predict(np.array([[300.0]]))[0]`.

Type

float

reg_

The linear regressor for $\ln k$ versus $\frac{1}{T}$.

Type

sklearn.linear_model.LinearRegressor

fit(*X*, *y*, *sample_weight=None*, ***kwargs*)

Fit the Arrhenius empirical equation.

Parameters

- **X** (*array-like of shape (n_samples, 1)*) – Temperature data.
- **y** (*array-like of shape (n_samples,)*) – Target values of the thermally-induced process.
- **sample_weight** (*array-like of shape (n_samples,), defualt=None*) – Individual weight of each thermally-induced value.
- ****kwargs** – Keyword arguments that are passed and are documented in `sklearn.linear_model.LinearRegression`.

predict(*X*)

Predict the thermally-induced process values.

Parameters

X (*array-like of shape (n_samples, 1)*) – Temperature data.

to_dataframe(*X*, *y*, *sample_weight=None*)

Convert the data with the predictions to a `pandas.DataFrame`.

Parameters

- **X** (*array-like of shape (n_samples, 1)*) – Temperature data.

- **y** (*array-like of shape (n_samples,)*) – Target values of the thermally-induced process.
- **sample_weight** (*array-like of shape (n_samples,), defualt=None*) – Individual weight of each thermally-induced value.

Returns
A `pandas.DataFrame` with the data.

Return type
pandas.DataFrame

plot(*ax=None*, *data_kws=None*, *pred_kws=None*)

Arrhenius plotter.

Parameters

- **ax** (*matplotlib.axes.Axes, default=None*) – The current axes.
- **data_kws** (*dict, default=None*) – Additional keyword arguments that are passed and are documented in `matplotlib.axes.Axes.errorbar` for the data points.
- **pred_kws** (*dict, default=None*) – Additional keyword arguments that are passed and are documented in `matplotlib.axes.Axes.plot` for the predictions values.

Returns
ax – The current axes.

Return type
matplotlib.axes.Axes

3.4 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX