

Linear Regression in materials science

– Correlations with melting temperature

In this module

- Introduction to linear regression with materials examples
 - Hands-on tutorials using nanoHUB: Young's modulus
- Hands-on tutorials using nanoHUB: correlating materials properties (this lecture)
 - Homework assignments

Saaketh Desai and Alejandro Strachan

desai61@purdue.edu || strachan@purdue.edu

School of Materials Engineering & Network for Computational Nanotechnology

Purdue University

West Lafayette, Indiana USA



Objective and prerequisites

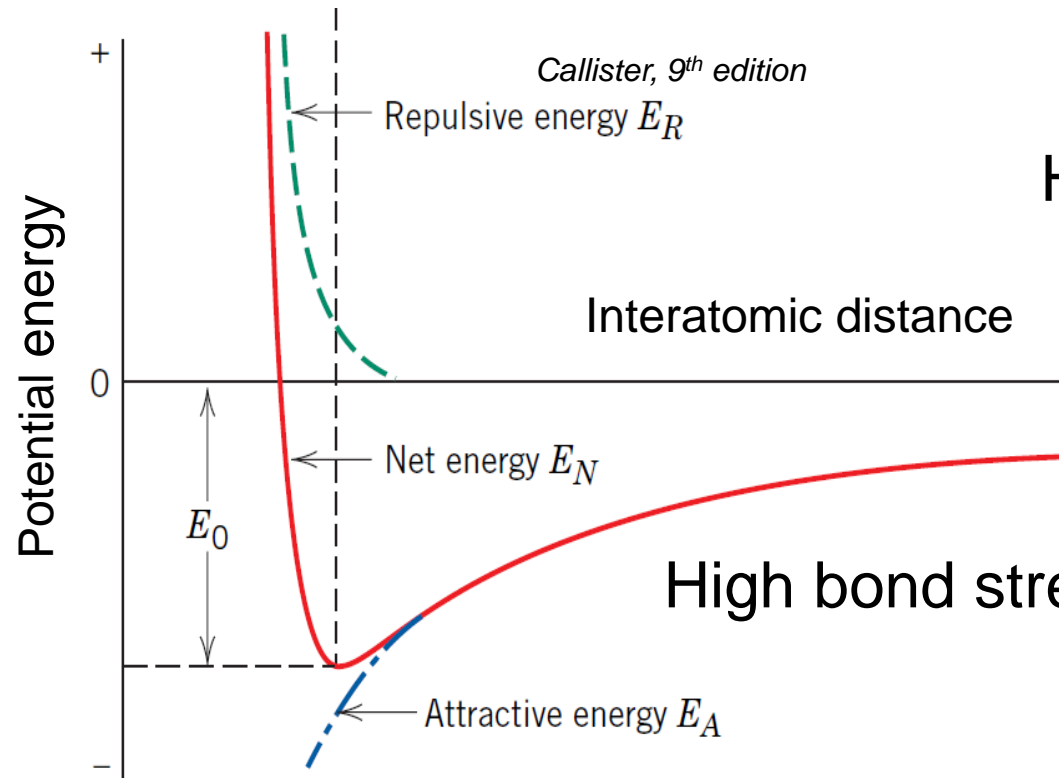
After completing this lecture you will:

- Learn to query materials dataset from an online database
- Construct and train a linear model to predict values from a materials dataset
- Evaluate uncertainties in the fitted parameters
- Compute errors in the fitting procedure

Pre-requisites:

1. Basic Python programming. See Juan C. Verduzco <https://nanohub.org/resources/33266>
2. Querying materials repositories (in this series)

Linear Regression – a materials example



High bond strength – high melting temperature

High bond strength – high stiffness – high Young's modulus

Melting temperature and Young's modulus generally correlate to each other

Can we predict Young's modulus based only on the melting temperature?

Step 1: Launching a Jupyter tool in nanoHUB

Machine Learning for Materials Science: Part 1

From your browser go to link: <https://nanohub.org/tools/mseml/>

Machine Learning for Materials Science: Part 1 Collect

By [Juan Carlos Verduzco Gastelum](#)¹, [Alejandro Strachan](#)¹, [Saaketh Desai](#)¹
1. Purdue University

Machine learning and data science tools applied to materials science

[Edit](#)

Launch Tool

Version 1.1 - published on 25 Feb 2019
doi:10.21981/9QJN-7N65 [cite this](#)
Open source: [license](#) | [download](#)

[View All Supporting Documents](#)

👤 1087 users, [detailed usage](#)
🗉 0 Citation(s)
💬 0 questions ([Ask a question](#))
★ 1 review(s)
👤 0 wish(es) ([New Wish](#))

→ Share: [f](#) [t](#) [s](#) ...

Click on Launch Tool to begin

Step 2: Landing Page – Notebook: Linear Regression

Navigate to the second link in the landing page to access the notebook we will be working on during this workshop

tools/mseml/bin/ x MSE_Machine_Lea x MSEML_LinearReg x MSEML_Regressio x MSEML_Classificati x https://prox

Not secure | proxy.nanohub.org/weber/1603167/GeOqVkJAVj2sIIN8J/11/notebooks/tools/mseml/bin/MSE_Machine_Learning_

nanoHUB jupyter MSE_Machine_Learning_Tutorials (autosaved)

File Edit View Insert Cell Kernel Widgets Help Snippets

+ ↻ ↺ ↻ ⬆ ⬇ ⬆ ⬇ Run ■ ↻ ⬆ ⬇ Markdown ▾ Appmode

Introduction to Machine Learning for Materials Science

The tutorials here will give you an insight into the usage of Machine Learning to approach problems related to mat

- **Get started** Click on the links below to begin each tutorial.
- **Important** To exit individual tutorials and return to this page, use File -> Close and Halt. "Terminate Session"

Querying databases, Organizing and Plotting Data:

- Query Pymatgen and Mendeleev for properties like Young's modulus and melting temperature
- Organize data into Pandas dataframes and python dictionaries and plot using Plotly

Linear Regression to predict material properties:

- Perform linear regression using the scikit learn package and predict Young's modulus
- Visualize trends in data and 'goodness of fit' of linear model

Neural Network Regression to predict material properties:

- Use neural networks to perform non-linear, higher order regression
- Visualize trends and compare non-linear model to linear regression

Neural Network Classification to predict crystal structures:

- Use neural networks to classify elements according to their crystal structures

Step 3: Let's get some data

1. Getting a dataset

```
import numpy as np
import pymatgen as pymat
import mendeleev as mendel
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from random import shuffle
import matplotlib.pyplot as plt

fcc_elements = ["Ag", "Al", "Au", "Cu", "Ir", "Ni", "Pb", "Pd", "Pt", "Rh", "Th", "Yb"]
bcc_elements = ["Ba", "Cr", "Eu", "Fe", "Li", "Mn", "Mo", "Na", "Nb", "Ta", "V", "W"]
hcp_elements = ["Be", "Ca", "Cd", "Co", "Dy", "Er", "Gd", "Hf", "Ho", "Lu", "Mg", "Re",
               "Ru", "Sc", "Tb", "Ti", "Tl", "Tm", "Y", "Zn", "Zr"]
others = ["Sb", "Sm", "Bi", "Ce", "Sn", "Si"]
# Others (Solids): "Sb", "Sm", "Bi" and "As" are Rhombohedral; "C", "Ce" and "Sn" are Allotropic;
# "Si" and "Ge" are Face-centered diamond-cubic;

elements = fcc_elements + bcc_elements + hcp_elements + others
```

Import scikit learn for linear regression

Query Pymatgen and Mendeleev for atomic number, melting point etc.

2. Processing and Organizing Data

```
for item in elements:
    data_youngs_modulus.append(pymat.Element(item).youngs_modulus)
    data_lattice_constant.append(mendel.element(item).lattice_constant)
    data_melting_point.append(mendel.element(item).melting_point)
    data_specific_heat.append(mendel.element(item).specific_heat)
    data_atomic_mass.append(pymat.Element(item).atomic_mass)
    data_CTE.append(pymat.Element(item).coefficient_of_linear_thermal_expansion)

# These would be the sets for Atomic Mass

mass_train = data_atomic_mass[:45]
mass_test = data_atomic_mass[-6:]
mass_train = np.array(mass_train).reshape(-1,1)
mass_test = np.array(mass_test).reshape(-1,1)

# These would be the sets for CTE

coefTE_train = data_CTE[:45]
coefTE_test = data_CTE[-6:]
coefTE_train = np.array(coefTE_train).reshape(-1,1)
coefTE_test = np.array(coefTE_test).reshape(-1,1)
```

Divide data into training and testing sets

- Training: used to parameterize the model
- Testing: check the model can be used as a stopping criteria – avoid overfitting

You can change the number of training/testing data here

Organize data into numpy arrays

Step 4: Create linear model and train model

3. Function to define model, train it and make predictions

```
def regression(x_train, x_test, y_train, y_test):  
  
    # Define the model and train it  
    model = linear_model.LinearRegression()  
    model.fit(x_train, y_train)  
  
    #Join train + test data  
    full_x = np.concatenate((x_train, x_test), axis=0)  
    full_y = np.concatenate((y_train, y_test), axis=0)  
  
    # Use the model to predict the entire set of data  
    predictions = model.predict(full_x) # Make it for all values  
  
    # Print model and mean squared error and variance score  
    print("Linear Equation: %.4e X + (%.4e)"%(model.coef_, model.intercept_))  
    print("Mean squared error: %.4e" % (mean_squared_error(full_y, predictions)))  
    print('Variance score: %.4f' % r2_score(full_y, predictions))  
  
    return predictions
```

Use scikit-learn to fit a linear model
Scikit-learn: <https://scikit-learn.org/stable/>

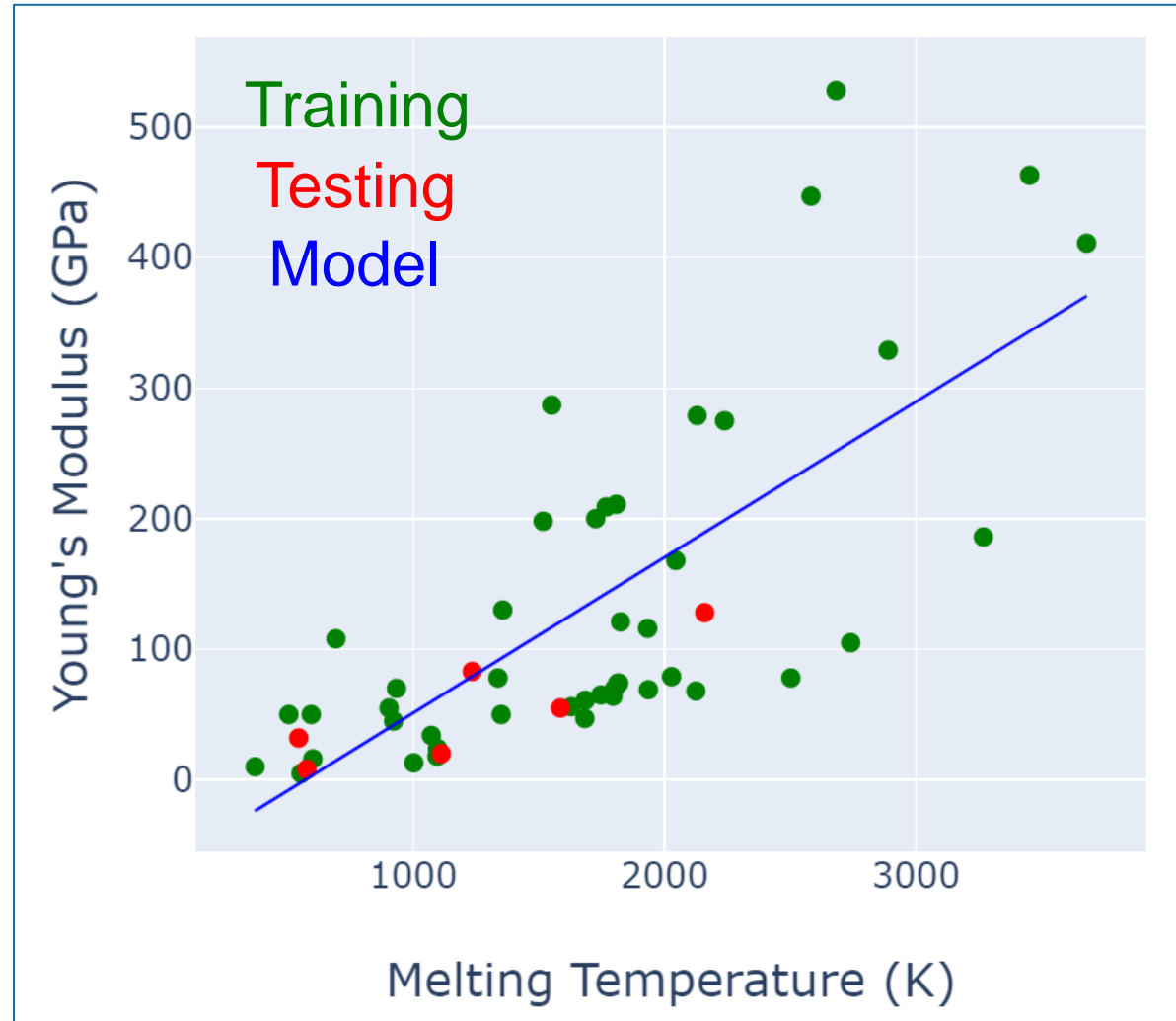
Print model parameters

5. Train models and plot results

```
predictions = regression(melt_train, melt_test, young_train, young_test)  
# This line calls the Regression model implemented in the function  
  
plot(melt_train, melt_test, young_train, young_test, "Melting Temperature (K)", "Young's Modulus (GPa)", predictions)  
# This line plots the results from that model
```

Plot results

Plot results



Next steps

Homework assignment: to reinforce concepts and help students modify the workflow and adapt it for their needs