Data Science and Machine Learning for MSE students: introduction and hands-on activities

Alejandro Strachan, Juan Carlos Verduzco and Saaketh Desai
School of Materials Engineering, Purdue University

August 2019

Abstract

This document introduces basic concepts of data science and machine learning in the context of materials science applications. The focus is on hands-on activities where readers use open, online tools in nanoHUB to explore the concepts being introduced. Topics covered include querying data resources and data organization and preparation. Regression exercises, including neural networks, to predict materials properties from a set of descriptors and a classification exercise designed to predict the crystal structure of elemental metals. The examples are provided as fully contained Jupyter notebooks and state of the art, open software. They are designed to introduce to the main steps in data science workflows and readers can modify or extend them to solve additional problems.

1 Introduction and overview

As in commerce, entertainment, auto, finance, marketing, and health care data science (DS), including machine learning (ML), is playing an increasingly important role in science and engineering.[1, 2, 3] Materials science is no exception [4, 5, 6]. The goal of DS is to extract actionable information out of data and ML is an important component of it that seeks to find patterns in data to create predictive models. A key feature of DS/ML is that information can be extracted without knowledge of the underlying mechanisms or physics behind the process of interest. This is attractive in many materials applications where only approximate models are known and they lack the accuracy or domain of applicability required. For example, predicting the thermodynamic stability [7, 8] and thermo-mechanical properties of alloys [9, 10], extracting microstructural features from a database of images [11], and developing interatomic potentials for molecular dynamics simulations using extensive databases of *ab initio* calculations [12, 13]. It is important to highlight that it is often the case in materials science that the amount of data is not enough to train physics-agnostic models. Thus, incorporating physics into machine learning models is not only exciting

from a basic science point of view but also of practical importance in many applications[14].

There are many excellent review and introductory articles on data science and machine learning [15], including several with a focus on MSE, see for example Refs. [16, 17]. The goal of this document and the accompanying resources is to complement this existing material with hands-on computational activities using open, online resources. In order to lower the barrier of access to DS/ML tools and jump start hands-on learning, we introduce a set of open, online resources that can used to explore DS and ML tools to address MSE challenges. *Machine Learning for Materials Science (MSEML)* is a collection of Jupyter notebooks walk users through the key steps utilized in the majority of DS/ML activities. The tools are open, free of charge, and build on state-of-the-art software and hardware. They are available through the US National Science Foundation's nanoHUB and the only requirement is a standard web browser, no need to download or install any software.

The MSEML tool, available at `https://nanohub.org/resources/mseml`, consists of four notebooks that exemplify tasks of increasing complexity. Users actually run the code and can modify it to explore new concepts, test their ideas and use them a blueprint to solve their own DS/ML problems. The initial steps in DS/ML workflows involve gathering and organizing data, the first n otebook in the MSEML series queries data repositories and libraries to gather information and the organizes it and visualizes it. Once the data is collected and organized it can be used to develop (or train in ML parlance) models, the second and third notebooks exemplify simple linear regression and deep neural networks regressions. Finally, the last notebook in the series exemplifies another important application of ML, classification.

The rest of the article is organized as follows. Section 2 describes the initial steps to acquire, organize and visualize data from online free databases for materials properties. Section 3 explains the linear regression algorithm widely used to model sets of data points. Section 4 builds on that and establishes some concepts of machine learning by adding a neural network that can take several descriptors for more accurate predictions. Section 5 walks through a classification problem in which elements are sorted by their crystallographic structure. Finally, section 6 explores clustering, and how to extract data from unlabeled examples.

2 Querying, organizing and visualizing data

The first step in DS/ML workflows is, of course, obtaining the data to be used and its organization in a manner that is suitable for analysis. In some cases all the data of interest is produced in house but it is often the case that models can be improved by including related data available in open or proprietary databases. Partially motivated by this, researchers are increasingly making their data available online via open repositories or libraries, using modern infrastructure to make it discoverable and mineable . Examples of successful open

repositories include the Materials Project[18], the Materials Data Facility [19], NIST, Citrination, AFLOW [20], most of which are complemented by libraries / APIs making them easier to mine.

To exemplify these initial steps the notebook *Querying databases, Organizing and Plotting Data* extracts data from two popular libraries and plots it. To create a dataset of elemental properties, we use Pymatgen [21] and Mendeleev [22]. Pymatgen (Python Materials Genomics) is an open-source, robust and well-documented library used for materials analysis, and is one of the codes powering the Materials Project. Mendeleev is a Python API linked to a database of atomic properties. These properties were obtained for a subset of elements and chosen to explore correlations between them and develop models to predict various properties.

Python offers great flexibility on how to store, sort and organize data with structures such as tuples, lists, arrays and libraries such as Pandas `https://pandas.pydata.org/` enable additional functionalities. Pandas is a python package created to provide users with flexible 2D structures that allow for easy indexing, sorting, data manipulation and data analysis. Each index represents an instance of a data point, complete with all its attributes. Pandas dataframes are utilized in the second notebook within MSEML that will be introduced next. This would be a good time for readers to launch the first notebook and run the first few cells up to the point where the first plot is created. Remember to hit Shift-Enter (or click on the top menu) to run each cell and to read the comments.

Python offers several visualization tools like Matplotlib[23] and Plotly[24] that are versatile and highly customizable and extensively by researchers. Matplotlib allows for a high degree of customization of the plots, while Plotly allows for interactive high-quality plots to explore correlations in data. The first notebook in MSEML introduced Plotly and subsequent notebooks incorporate Matplotlib.

It is often the case that the raw data must be pre-processed and enhanced prior to being used to generate any kind of prediction. Models are susceptible to differences in the magnitudes of the descriptors that sway the fitted mathematical model to better adjust for greater quantities.

2.1 Suggested activities

The notebook includes several suggested activities, readers are encouraged to perform them as they will help with the activities in the following sections.

3 Supervised learning: linear regression

Regression is a statistical method to find relationships between variables and linear regression is its simplest manifestation. The notebook *Linear Regression to predict material properties* finds the relationship between the Young's

modulus and melting temperature for various metallic elements using a standard least-squares linear regression implementation within the scikit-learn[25] library.

This example uses a least-squares regression where the optimal function minimizes the sum of the squares of the difference between the predicted and actual values:

$$\sum_i \left(f(x_i) - y_i\right)^2 \tag{1}$$

where y_i is the output corresponding to the input x_i. The excellent review by Mueller et al. is recommended for anyone interested in further exploring regression. In linear regression the optimal slope m and a intercept b are obtained as follows:

$$m = \frac{N * \sum(xy) - \sum(x) * \sum(y)}{N * \sum(x^2) - \sum(x)^2} \tag{2}$$

$$b = \frac{\sum(y) - m * \sum(x)}{N} \tag{3}$$

where N is the number of data points.

Establishing models for materials properties with linear regression presupposes that a relationship between the independent variables exists and that the model can span the entire range of the inputs.[26]

3.1 Suggested activities

Readers can modify the Linear Regression notebook to explore correlations between all the possible pairs of properties from the queried databases. A suggested activity is to find the pair of properties that shows the strongest linear. Both being dominated by interatomic bonding, one could intuitively expect correlation between melting temperature and stiffness. However, it is interesting to explore possible correlations between seemingly different properties.

A function provided within the notebook allows for easy generation of comparison plots between properties and provides the user with the values for the sum of the residual error and the equation of the best fit we discussed prior.

4 Supervised learning: neural network regression

Section 3 showed that there are important correlations between materials properties. For example, materials with high melting temperatures tend to be stiffer (i.e. have a higher Young's modulus). At the same time, there is significant spread in the data around the linear model developed. Consequently, if one is interested in predicting the stiffness of a metal of known melting temperature, one would make, in average, quite a large error. In this next exercise, we explore

more flexible models capable of predicting materials properties, specifically we will use neural networks to predict the Young's modulus of the elemental materials above from a variety of descriptors, including periodic table data and other properties. This is done in the third Notebook in MSEML: *Neural Network Regression to predict material properties*.

Neural networks and deep neural networks are very popular models in machine learning because they are universal approximators, i.e., in principle, they can describe any continuous function between inputs and outputs [27]. It is important to note that often the size of the network required for a given problem is known a prior, nor how the ideal network parameters can be learned. A neural network is a group of interconnected artificial neurons that relate inputs (in our case the descriptors of the materials of interest) to one or multiple outputs (in our case the Young's modulus). Each neuron performs a very simple task, it takes one of more inputs and produces a single output that is communicated to one or multiple neurons or represents an output of the network. Adjustable *weights* control the strength of the connection between neurons and each neuron adds all its inputs and an *activation functions* relate this total input to the output. To learn more about artificial networks see Ref. [28] and references therein.

Just as before, the first few cells of the notebook retrieve data from Mendeleev and Pymatgen and organize into a Pandas dataframe. Following that, the data is shuffled and divided into training and testing sets, the testing data will not be used to train the models. The input and output data is then centered and normalized, this is important as different categories of input data can have vastly different and unit dependent values (e.g. atomic mass and melting temperature). At this point readers an encouraged to run Steps 1 and 2 of the network.

Once the data is prepared, step 3 in the notebook creates of a feed-forward network. Keras makes it very easy to build such model, for each layer we specify how many neurons we want, their activation functions and its initial weights. The first layer also determines the size of the input space. We use the Rectified Linear Unit (ReLU) activation but this can be changed to many other options, see `https://keras.io/activations/`. Readers should create the network and explore the summary description.

After creating the desired network, we are ready to train it with the data at hand. As we setup the training we specify a *validation split* of 0.1. This means that the training data will be further divided, 90% of the data will be used for training and the rest will be used for validation; this validation set is important during training to assess and avoid overfitting. We also specify the number of epochs (iterations over all samples) that will be used during training, where weights are adjusted with forward and backpropagation. In the default example, the training is performed for a fixed number of *epochs* as opposed to monitoring for the error in the validation set, which is left as a exercise for the reader. Finally, the model is evaluated on the testing set and the predictions of the model are plotted vs. the actual values.

4.1 Suggested activities

1. After setting up the network we print a summary of the network with information about each layer. Explain the number of adjustable parameters in each layer.

2. Modify the network to use the error in the validation set as the stopping criterion. See https://keras.io/callbacks/#earlystopping.

3. Train the network using only atomic properties as descriptors (i.e. no information about the solid) and characterize the RMS error. Then, add properties such as melting temperature and cohesive energy that provide information about the solid; this can be though of as adding physics to the network as one can expect these properties to correlate better with the Young's modulus. Train the network with the additional descriptors and compare the accuracy of the models.

4. Develop a model for melting temperature instead of Young's modulus.

5 Supervised learning: classification

Some of the earliest and most popular examples of machine learning involved image recognition tasks, starting from the classic examples of classifying handwritten digits [29] to classifying millions of images across different objects using specialized neural network architectures such as Convolutional Neural Networks [30]. In the materials domain, classification is used, for instance, to predict crystal structures alloys with novel compositions [31, 32, 33]. As an introductory example, we developed a Jupyter notebook, *Neural Network Classification to predict crystal structures*, that mines the Pymatgen and Mendeleev databases for single element properties and train a standard feedforward neural network to predict the ground state crystal structures of these elements.

As discussed above, the first step is to extract features from the database of material properties. In this case, we directly use properties present in the database, such as atomic radius, lattice constant, heat of formation etc. as features, or descriptors, of our single element. As will be discussed in the Suggested activities, one could pre-process the data using methods such as principal component analysis (PCA) or Pearson/Spearman correlations to perform feature reduction and identify a minimal set of descriptors required to identify the correct crystal structure.

The descriptors for each element are organized into a Pandas dataframe, along with the corresponding crystal structure labels (FCC, BCC or HCP) from the database, to form our training set containing 47 training examples. We then use the Keras sequential model API to define a simple, densely connected, feedforward neural network with one hidden layer of 16 neurons. The output layer consists of 3 neurons, the values of which will determine the probability associated with that crystal structure. Finally, the crystal structure with the maximum probability is reported as the network's predicted crystal structure.

At this point, readers are encouraged to run the notebook up to the definition of the neural network.

A key distinction in using neural networks for classification as opposed to regression is the use of a 'softmax' activation function for the output layer as opposed to a linear activation. The reason is that while a regresssion task requires the prediction of a continuous, unbounded value (only possible with a linear activation function since other activation functions restrict the output range), a 'softmax' activation function returns values that are always between 0 and 1, such that the n outputs of the layer always add to 1, thus allowing us to interpret these values as probabilities.

The other distinction in training neural networks to perform classification is to use an appropriate error metric, typically a classification accuracy, as opposed to a mean-squared error metric typically used for regression.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of total predictions}} \tag{4}$$

In conjunction, the objective function used to train the network is now a 'categorical cross-entropy' function.

$$\text{Cross-Entropy Loss} = \sum_{i=1}^{C} y_i' log(y_i) \tag{5}$$

$$y_i = \frac{exp(x_i)}{\sum_{j=1}^{C} exp(x_j)} \tag{6}$$

where C is the number of categories and y_i' is the true probability of that category.

In our case, this simple network can predict the crystal structures of single elements (either FCC, BCC or HCP) with $\sim$ 90 % accuracy on the training set and $\sim$ 70 % accuracy on the test set. For other crystal structures or multi-component alloys, we refer to the literature above as starting points for training your own models.

5.1 Suggested activities

In our simple example to predict single element, ground state crystal structures, we use 18 features including features you would expect to be highly correlated, such as evaporation heat and melting / boiling point, as well as features that would generally be uncorrelated, such as electrical resistivity and melting point. To explore this, you can use the Pearson or Spearman correlation functions from Scipy and remove features that show high correlations with other features. Alternately, you could use a Principal Component Analysis (PCA) decomposition from scikit-learn of the feature space, generating a smaller set of equivalent features that explain a large fraction of the data (note that our small dataset might result in some artificial correlations, but the techniques hold good in general).

Algorithms such as decision trees or random forests (collection of decision trees) inherently rank features according to their importance in classifying accurately. Once again, you could use the scikit-learn package to fit a random forest model on the raw dataset, observing which features were learnt to have high importance, and maybe even compare them to the ones you manually selected from the correlation coefficients, or the ones chosen by a PCA technique.

Neural networks and decision trees can be prone to overfitting, particularly for small datasets such as ours. You could use Support Vector Machines (SVMs), another widely used classification technique, to alleviate this problem. Scikit-learn again provides a useful comparison of various classifiers, including SVMs, with starter code.

6 Unsupervised learning

Coming soon.

7 Conclusions

Data science and machine learning are becoming central tools in science and engineering; this document introduced a set of online tools designed to introduce some of the basic concepts in the field with examples of interest in the field of materials. The goal is to jump start learning with hands-on activities that readers can run, modify and extend using a standard web browser. We believe that sharing examples via nanoHUB is a powerful avenue to train undergraduate and graduate students as well as working professionals. We encourage readers with interesting modifications or extensions of these tools or authors of completely independent notebooks to share their work with their comity by publish their own tools in nanoHUB. The process of publication is very simple and most steps are automated, see `https://nanohub.org/whypublish/` and all nanoHUB tools receive a unique digital object identifier and are indexed by Web of Science and Google Scholar.

References

[1] Mukta Paliwal and Usha A. Kumar. Neural networks and statistical techniques: A review of applications. *Expert Systems With Applications*, 36(1):2–17, 2009.

[2] Connorw. Coley, Wengong Jin, Luke Rogers, Timothy F. Jamison, Tommi S. Jaakkola, William H. Green, Regina Barzilay, and Klavs F. Jensen. A graph-convolutional neural network model for the prediction of chemical reactivity. *Chemical Science*, 10(2):370–377, 2019.

[3] Ratiranjan Jena, Biswajeet Pradhan, Ghassan Beydoun, Hizir Nizamuddin, Muzailin Ardiansyah, Muzailin Sofyan, and Muzailin Affan. Integrated

model for earthquake risk assessment using neural network and analytic hierarchy process: Aceh province, indonesia. *Geoscience Frontiers*, 2019.

[4] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715), 2018.

[5] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[6] National Science and Technology Council (US). *Materials genome initiative for global competitiveness*. Executive Office of the President, National Science and Technology Council, 2011.

[7] Bryce Meredig, Ankit Agrawal, Scott Kirklin, James E Saal, JW Doak, Alan Thompson, Kunpeng Zhang, Alok Choudhary, and Christopher Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[8] Geoffroy Hautier, Christopher C Fischer, Anubhav Jain, Tim Mueller, and Gerbrand Ceder. Finding nature's missing ternary oxide compounds using machine learning and density functional theory. *Chemistry of Materials*, 22(12):3762–3767, 2010.

[9] Atsuto Seko, Tomoya Maekawa, Koji Tsuda, and Isao Tanaka. Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single-and binary-component solids. *Physical Review B*, 89(5):054303, 2014.

[10] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2:16028, 2016.

[11] Brian L DeCost, Bo Lei, Toby Francis, and Elizabeth A Holm. High throughput quantitative metallography for complex microstructures using deep learning: a case study in ultrahigh carbon steel. *Microscopy and Microanalysis*, 25(1):21–29, 2019.

[12] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010.

[13] Aidan P Thompson, Laura P Swiler, Christian R Trott, Stephen M Foiles, and Garritt J Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015.

[14] Ekin D. Cubuk, Austin D. Sendek, and Evan J. Reed. Screening billions of candidates for solid lithium-ion conductors: A transfer learning approach for small data. *The Journal of Chemical Physics*, 150(21), 2019.

[15] M I Jordan and T M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science (New York, N.Y.)*, 349(6245), 2015.

[16] T Mueller, AG Kusne, and R Ramprasad. Machine learning in materials science: Recent progress and emerging applications. *Reviews In Computational Chemistry, Vol 29*, 29:186–273, 2016.

[17] Weike Ye, Chi Chen, Shyam Dwaraknath, Anubhav Jain, Shyue Ping Ong, and Kristin A. Persson. Harnessing the materials project for machine-learning and accelerated discovery. *MRS Bulletin*, 43(9):664–669, 2018.

[18] G. Hautier W. Chen W.D. Richards S. Dacek S. Cholia D. Gunter D. Skinner G. Ceder K.A. Persson A. Jain, S.P. Ong. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[19] K. Chard J. Pruyne R. Ananthakrishnan S. Tuecke Blaiszik, B. and I. Foster. The Materials Data Facility: Data services to advance materials science research. *JOM*, 68(8):2045–2052, 2016.

[20] Stefano Curtarolo, Wahyu Setyawan, Gus LW Hart, Michal Jahnatek, Roman V Chepulskii, Richard H Taylor, Shidong Wang, Junkai Xue, Kesong Yang, Ohad Levy, et al. Aflow: an automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012.

[21] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L. Chevrier, Kristin A. Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68(C):314–319, 2013.

[22] Łukasz Mentel. mendeleev – a python resource for properties of chemical elements, ions and isotopes.

[23] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[24] Plotly Technologies Inc. Collaborative data science, 2015.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Hkdh Bhadeshia. Neural networks in materials science. *Isij International*, 39(10):966–979, 1999.

[27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[28] A.K Jain, Jianchang Mao, and K.M Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996.

[29] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[31] Prasanna V Balachandran, Antoine A Emery, James E Gubernatis, Turab Lookman, Chris Wolverton, and Alex Zunger. Predictions of new ab o 3 perovskite compounds by combining machine learning and density functional theory. *Physical Review Materials*, 2(4):043802, 2018.

[32] Angelo Ziletti, Devinder Kumar, Matthias Scheffler, and Luca M Ghiringhelli. Insightful classification of crystal structures using deep learning. *Nature communications*, 9(1):2775, 2018.

[33] Woon Bae Park, Jiyong Chung, Jaeyoung Jung, Keemin Sohn, Satendra Pal Singh, Myoungho Pyo, Namsoo Shin, and K-S Sohn. Classification of crystal structure using a convolutional neural network. *IUCrJ*, 4(4):486–494, 2017.