Lecture 0: Machine learning in Chemistry 101

Bingqing Cheng, Lucy Colwell

January 2020

Description

This graduate-level course gives an overview of the machine learning (ML) techniques that are useful for solving problems in Chemistry, and particularly for the computational understanding and predictions of materials and molecules at the atomistic level.

In the first part of the course, after taking a quick refresher of the basic concepts in Probabilities and Statistics, students will learn about basic and advanced ML methods including supervised learning and unsupervised learning. During the second part, the connection between Chemistry and the mathematical tools of ML will be made, and the concepts on the construction of loss functions, representations, descriptors and kernels will be introduced. For the last part, experts who are actively using ML methods to solve research problems in Chemistry and Materials will be invited to give real world examples on how did ML methods transformed the way they perform research.

Timetable

Lent term 2020, Jan/Feb 9+ lectures x 50 min + 2 workshops x 3 hours

Prerequisites

- Introductory knowledge of statistical mechanics and probability, basic programming skills in Python.
- The students will be given some notes on how to install Jupyter notebooks on their laptops before the course, or how to use them via Google Colab.

Computation

Students are encouraged to work in Python, and Python notebooks will be provided for the workshops. No formal background in python is required, although previous knowledge of any programming language will be an advantage. Course materials will be provided via the course Moodle site, available at https://www.vle.cam.ac.uk/course/view.php?id=73991

Bibliography

- A high-bias, low-variance introduction to Machine Learning for physicists https://physics.bu.edu/ pankajm/MLnotebooks.html
- David Mackay: Information theory inference and learning algorithms
- Christopher M. Bishop: Pattern Recognition and Machine Learning
- Aurélien Géron: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition
- Peter Eastman; Vijay Pande; Patrick Walters; Bharath Ramsundar: Deep Learning for the Life Sciences

Notations used in this course (and often elsewhere)

One-dimensional data We use the small letter x to denote an *independent variable* (also called a *feature*, an *input*, a *predictor*, or an *explanatory variable*) in one dimension, which is just a scalar.

One-dimensional labelled data If the one dimensional data point has a label y, which can be the name or the tag of a class or a continuous observable, we write the data point as (y, x). y is often referred to as a *dependent variable*, an *output*, a *target* a *realization*, or an *outcome*.

One-dimensional data set Frequently, one has to deal with a data set with many data points. In that case, we use the form $\{x_{i=1\ldots N}\} \equiv (x_1, \cdots, x_N)^T$ to denote an one-dimensional data set, where the index $i = 1 \ldots N$ runs over the number of samples. If the data set is also labelled, then we use $\{(y_{i=1\ldots N}, x_{i=1\ldots N})\}$.

High-dimensional data If a data point lives in more than one dimension, i.e. it has a feature space of d-dimension, we use $\mathbf{x}^d$ to express it. $\mathbf{x}^d \in \mathcal{R}^d$, in essence, is a feature (row) vector, i.e. a matrix that has a row column of d elements with index $j = 1 \ldots d$:

$$\mathbf{x}^d = \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots & x^{(d)} \end{bmatrix}. \tag{1}$$

We often omit the superscript d if the dimensionality is not important or remain constant in a particular scenario. If the data is labelled, we can call it $(y, \mathbf{x})$.

High-dimensional data set Similarly, for a d-dimensional data set with N samples, we use $\{\mathbf{x}_{i=1\ldots N}\}^d \equiv (\mathbf{x}_1, \cdots, \mathbf{x}_N)^T$. If labelled, we use $\{(y_{i=1\ldots N}, \mathbf{x}_{i=1\ldots N})\}^d$ or simply $\{(y_i, \mathbf{x}_i)\}$, where the index $i = 1 \ldots N$ runs over the number of examples and $\mathbf{x}_i$ is a d-dimensional feature (row) vector.

Design matrix To express a high-dimensional data set in a clean and standard way, we use the *design matrix*, also known as *model matrix* or *regressor matrix* and often denoted by $\mathbf{X}$. The *design matrix* is a $N \times d$ matrix whose rows, $\mathbf{x}_1, \cdots, \mathbf{x}_N$, are the examples and columns, $\mathbf{X}_{:,1}, \cdots, \mathbf{X}_{:,d}$, are the measured features. To write it down explicitly:

$$\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)^T = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_N \end{bmatrix}. \tag{2}$$

We also denote the dependent variables y of the N samples as a column vector $\mathbf{y} \equiv (y_1, \cdots, y_N)^T$. As such, the data set, often called the training set, can be expressed very compactly as $(\mathbf{y}, \mathbf{X})$.

Models and parameters Mathematically speaking, a prediction model is just a function $f(\mathbf{x}; \mathbf{w})$ that takes the independent variables $\mathbf{x}$ as inputs. The parameters of the model $\mathbf{w} \in \mathcal{R}^p$ are often called *regression parameters*, which are p-dimensional column vector: $\mathbf{w} \equiv (w_1, \cdots, w_p)^T$.

Feature space and basis functions Oftentimes it is advantageous to transform the raw inputs $\mathbf{x}$ into a *feature space* (or *basis*) using functions $\phi(\mathbf{x})$, which can then be used as inputs for regression, classification and prediction purposes. For a set of examples $\{\mathbf{x}_{i=1 \ldots N}\}$, we have a set of vectors in the feature space $\{\phi(\mathbf{x})_{i=1 \ldots N}\}$.