Computational Methods - Lesson 10 (Luxembourg, 02.12.2020)

Machine Learning I

Valentin Vassilev Galindo[*,a], Matteo Barborini[*,b], Prof. Alexandre Tkatchenko[*]

Theoretical Chemical Physics group

Faculté des Sciences, des Technologies et de Médecine - Département Physique et sciences des matériaux

[*]Campus Limpertsberg, Université du Luxembourg - 162 A, avenue de la Faïencerie, L-1511 Luxembourg

[a]valentin.vassilev@uni.lu, [b]matteo.barborini@uni.lu

1. Introduction to ML

1.1. Definition. Machine learning (ML) is the science of getting computers to act without being explicitly programmed, relying on patterns and inference instead. In the most general sense, ML algorithms recover functional relationships $\hat{f} : \mathcal{X} \xrightarrow{\text{ML}} \mathcal{Y}$ between inputs $\mathcal{X}$ and outputs $\mathcal{Y}$ using statistical means. Usually, those models are so-called *universal approximators* that can reconstruct any function that fulfils a few basic properties, such as continuity and smoothness.

1.2. Types of Learning.

1.2.1. Supervised Learning. Addresses learning problems where the ML model connects a set of known inputs and outputs, either to perform a regression or classification task.

- **Regression:** The goal is to predict a continuous number, or a floating-point number in programming terms (example: "To set the price of a house knowing its surface, number of rooms, and criminal rate in the neighbourhood")
- **Classification:** The goal is to predict a class label, which is a choice from a predefined list of possibilities (example: "Label a set of pets' pictures as birds, dogs and cats")

1.2.2. Unsupervised Learning. Describes less well defined problems with the goal to extract knowledge from inputs, without the target variables being known.

- **Clustering:** Aims to separate groups with similar traits and assign them into clusters (example: "Split up similar clothing into stacks")
- **Association:** The goal is to identify sequences within the data (example: "Find which clothes someone uses together")
- **Dimension reduction:** The goal is to find hidden dependencies (example: "Find the best outfits from the given clothes")

1.2.3. Reinforcement Learning. Describes problems that combine aspects of supervised and unsupervised learning. Reinforcement learning problems often involve defining an *agent* within an environment that learns by receiving feedback in the form of punishments and rewards. An example could be an ML algorithm that learns how to play a game (e.g. chess, the snake game, go, etc.).

1.3. Datasets. On a fundamental level, ML models are simply sophisticated parametrizations of datasets. The reference dataset ultimately determines how effective a model can be. If the dataset is not representative of the problem at hand, the model will be incomplete and behave unpredictably in situations that have been improperly captured. The same applies to biases or noise artifacts, which will also be reflected in the model.

Datasets are split into two subsets, a training set (in-sample) and a test set (out-of-sample). The former is used to train the model, and allows the algorithm to "learn" the problem, and the latter is employed to assess how good is the trained model. Sometimes there is a third subset called validation set, which is normally used to select the model with the set of hyperparameters (see below) with the best prediction accuracy.

1.4. Features. Features are parameters or variables. A feature is an individual measurable property or characteristic of a phenomenon being observed, which has some specific characteristic(s) to allow algorithms to work properly.

Features can be as simple as lengths, age of a person, surfaces, a color, or, in the case of a function *f(x)* the feature could simply be the value of *x*. However, one might have more complex features such as those found in character recognition (histograms counting the number of black pixels along horizontal and vertical directions, number of internal holes, stroke detection, etc.), speech recognition (noise ratios, length of sounds, relative power, etc.) or image processing (specific structures in the image such as points, edges or objects; motion in image sequences, or shapes defined in terms of curves or boundaries between different image regions).

1.5. Bias and Variance[1]. The bias represents the best our model could do if we had an infinite amount of training data. The bias is a property of the kind of functions, or model class, we are using to approximate *f(x)*. In general, the more complex the model class we use, the smaller the bias. However, we do not generally have an infinite amount of data. For this reason, to get best predictive power it is better to minimize the out-of-sample error (E$_{out}$), rather than the bias.

E$_{out}$ can be naturally decomposed into a bias, which measures how well we can hypothetically do in the infinite data limit, and a variance, which measures the typical errors introduced in training our model due to sampling noise from having a finite training set. One of the lessons of statistical learning theory is that it is not enough to simply minimize the training error, because the out-of-sample error can still be large (Fig. 1).

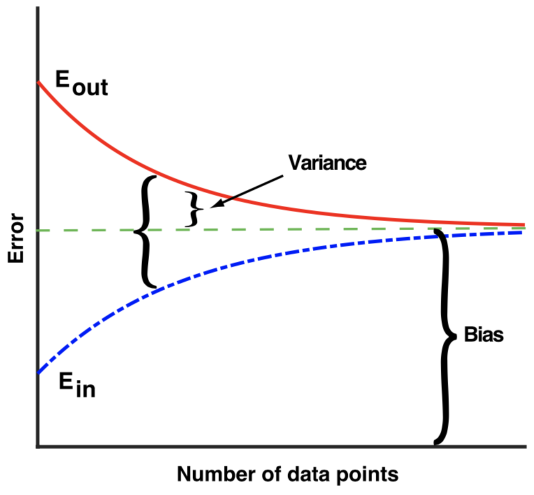

Figure 1: Schematic of typical in-sample and out-of-sample error as a function of training set size. The typical in-sample or training error (E$_{in}$), out-of-sample or generalization error (E$_{out}$), bias, variance, and difference of errors as a function of the number of training data points. The schematic assumes that the number of data points is large, and that our model cannot exactly fit the true function *f(x)*. Image taken from Mehta, P., *et al.*, *Physics Reports* **810** (2019), 1-124.

Model complexity is a very subtle idea and defining it precisely is one of the great achievements of statistical learning theory. In many cases, model complexity is related to the number of parameters we are using to approximate the true function *f(x)*. If we consider a training dataset of a fixed size, E$_{out}$ will be a non-monotonic function of the model complexity, and is generally minimized for models with intermediate complexity. The underlying reason for this is that, even though using a more complicated model always reduces the bias, at some point the model becomes too complex for the amount of training data and the generalization error becomes large due to high variance. This leads to the important concept commonly called the bias–variance trade-off, which gets at the heart of why machine learning is difficult (Fig. 2).

1.6. Model Selection. It is important to know that one needs to be really careful when selecting a trained model. Fig. 3 shows two types of issues that one needs to avoid when training a ML algorithm: Overfitting and

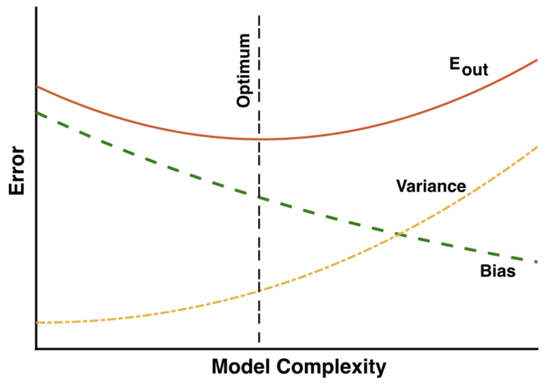

Figure 2: Bias–Variance trade-off and model complexity. This schematic shows the typical out-of-sample error E$_{out}$ as function of the model complexity for a training dataset of fixed size. Notice how the bias always decreases with model complexity, but the variance, i.e. fluctuation in performance due to finite size sampling effects, increases with model complexity. Thus, optimal performance is achieved at intermediate levels of model complexity.Image taken from Mehta, P., *et al.*, *Physics Reports* **810** (2019), 1-124.

underfitting.

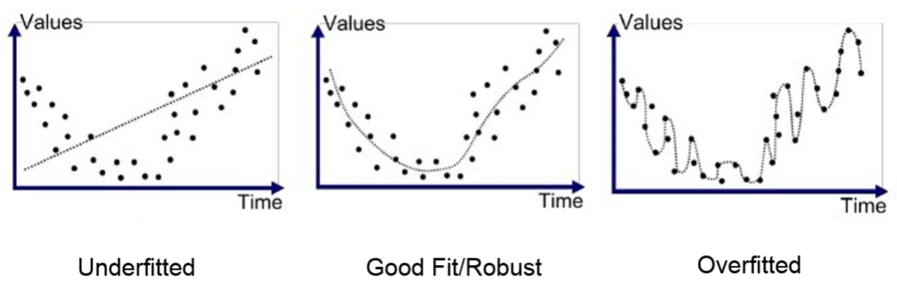

Figure 3: Examples of an underfitted, good fitted, and overfitted ML models. Points represent the training examples and the dashed lines the prediction of the ML model.

1.6.1. Loss function. In order to measure whether the prediction model f with parameters Θ is accurately reflecting the data, we need to define an appropriate loss function which is a function of the parameters Θ. For regression problems a common loss is the mean squared error (MSE)

$$\mathcal{L}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\mathbf{X}_i, \Theta) - y_i\right)^2 . \tag{1}$$

1.6.2. Hyperparameters. The hyperparameters are all the parameters which can be arbitrarily set by the user before starting training and they are not directly learnt within the estimators. Hyperparameters define higher level concepts about the model such as complexity, or capacity to learn. They are normally selected by setting different values, training different models, and choosing the values that test better. Some ML methods have several hyperparameters, while others have none.

2. Gradient Descent[1]

Almost every problem in ML and data science starts with the same ingredients: a dataset ***X***, a model *f(**X**, Θ)*, which is a function of the parameters Θ, and a loss function $\mathcal{L}$ (Θ) that allows us to judge how well the model *f(**X**, Θ)* explains the observations ***X***. The model is fit by finding the values of Θ that minimize the cost function. The basic idea behind Gradient Descent (GD) methods is straightforward: iteratively adjust the parameters Θ in the direction where the gradient of the loss function is large and negative. In this way, the training procedure ensures the parameters flow towards a local minimum of the loss function.

The underlying reason why training a ML algorithm is difficult is that the loss functions we wish to optimize are usually complicated, rugged, non-convex functions in a high-dimensional space with many local minima. To make things even more difficult, we almost never have access to the true function we wish to minimize: instead, we must estimate this function directly from data.

2.1. Gradient Descent in a Nutshell.

2.1.1. Setting Up Gradient Descent. To begin using GD one must define a hypothesis (i.e. the functional form we want *f(**X**$_i$, Θ)* to have). For the sake of simplicity, it would be:

$$f(\mathbf{X}_i, \Theta) = \Theta_0 X_{i,0} + \Theta_1 X_{i,1} + \Theta_2 X_{i,2} + \cdots + \Theta_M X_{i,M} \tag{2}$$

Where Θ is a vector containing all the parameters (Θ_0,Θ_1,...,Θ_M) that need to be "learnt" and **X**$_i$ is a vector whose elements are the values of the features of the ith observation ($X_{i,0}$,$X_{i,1}$,...,$X_{i,M}$). Please note that $X_{i,0}$ is only written for practical reasons and it is always equal to 1.
The second ingredient for GD is the loss function. For this lecture, we selected the equation of the MSE introduced in equation 1

$$\mathcal{L}(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \left(f(\mathbf{X}_i, \Theta) - y_i\right)^2 . \tag{3}$$

The use of 1/2N instead of 1/N is just a tool for simplifying the mathematical expressions used in GD. Please note that this will not affect the result of the algorithm.
The final goal of GD is to minimize the function $\mathcal{L}$ (Θ) by optimizing the values in Θ.

2.1.2. Feature Scaling. Feature scaling is an important preprocessing step in most ML algorithms. It is performed to handle highly varying magnitudes, values or units. If feature scaling is not done, then a ML algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values. Furthermore, it makes the optimization process faster. A typical scaling approach is the so-called min-max normalization

$$X_{i,j}^{New} = \frac{X_{i,j} - Min(\mathbf{X}_j)}{Max(\mathbf{X}_j) - Min(\mathbf{X}_j)} \tag{4}$$

2.1.3. Gradient Descent Algorithm.

1. Scale all feature vectors **X**$_j$ and the target vector **y** of the training examples. Remember to keep the values of Min(**X**$_j$), Max(**X**$_j$), Min(**y**) and Max(**y**), so that it would be possible to transform back any value to its original scale.
2. Initialize the parameters Θ with random values and calculate the value of the loss function.
3. Calculate the gradient (i.e. the first derivative) of the loss function for all values in Θ

$$\frac{\partial}{\partial \Theta_j} \mathcal{L}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \left(f(\mathbf{X}_i, \Theta) - y_i\right) X_{i,j} \tag{5}$$

4. Adjust the parameters with the gradients to reach the optimal values where the loss function is minimized

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} \mathcal{L}(\Theta) \tag{6}$$

   where α is the learning rate (i.e. how fast we go down the slope). One needs to update all parameters simultaneously.

5. Use the new parameters to calculate the new value of the loss function.
6. Repeat steps 3,4 and 5 until convergence (i.e. the value of the loss function does not change significantly compared to the previous step).

3. References

1. Mehta, P., *et al.*, *Physics Reports* **810** (2019), 1-124.