Machine Learning Introduction to Materials Science

Cibrán López Álvarez and Pol Benítez Colominas

European Kesterite Workshop
July 2023

Contents

1 Introduction

1.1 AI taxonomy

Machine learning (ML) started back in the fifties with Bayes theorem, Markov chains, etc. and it has been widely used lately in many fields: Netflix uses AI for recommending new films, some cars image detection to identify obstacles, etc.

However, it was just a few years ago that bumped into materials science. But let's start from the beginnig.

AI is a broad concept, which consists on these algorithms designed to perform tasks normally requiring human intelligence (eg, symbolic logic).

ML is based on these algorithms that learning and adapt through experience/data to complete specific tasks (eg, clustering algorithms). ML is a powerful tool for making predictions and decisions based on data. By training models on large datasets, we can extract insights and make predictions that would be impossible for humans to do manually. As we continue to develop more advanced models and techniques, the potential applications for machine learning are endless.

DL refers to these algorithms that directly try to mimic brain functioning (eg, convolutional neural networks). DL involves training neural networks with many layers, and it has revolutionized fields like computer vision and natural language processing, allowing computers to perform tasks like image classification and language translation with unprecedented accuracy.

1.2 ML taxonomy

ML can perform many different tasks, although it is necessary to select the correct approach:

- Classification: is this type A or B?
- Anomaly detection: is this weird?
- Regression: how much/many?
- Clustering: how is this organized?
- Reinforcement learning: what should I do next?
- ...

Broadly, we can divide it in three main techniques: supervised, semisupervised and unsupervised learning.

Supervised learning involves training a model on a labeled dataset, where each data point (atomic coordinates of an unit cell, for instance) is labeled with the correct output (photoabsorption, formation energy...). For example, we might train a model to predict the absorption of a new compound by showing it a large dataset of compounds with known absorptions.

Semisupervised learning includes in this labeled dataset some unlabeled points, for which we do not have information, but we expect this to be similar to close data. For example, in the dataset from the previous problem, we could include elements with isolectronic substitutions, and let the model assume that their absorption would be similar to the material for which we have data.

Unsupervised learning, on the other hand, involves training a model on a completely unlabeled dataset, where the model has to discover patterns and structure in the data itself. This is useful for tasks like clustering, where we want to group similar data points together. For example, we could give the dataset of all materials, and cluster them in terms of similarity, so we could assume elements from the same cluster might share properties such as absorption.

Then, the integration of ML into materials science has the potential to revolutionize the field, as it can be used to analyze large volumes of data, make predictions and optimize materials for specific applications.

2 Quality metrics

As we have seen, when approaching a new problem, we have to think about what we want to achieve (a number, a classification...) and what type of data we have (labeled, unlabeled or both).

Once we have defined that, we proceed to train the model. There are two (not totally uncorrelated) main topics we have to care about when training a model:

- Features: descriptors of the dataset.
- Fitting: training of the model given a dataset.

The first item refers to the construction of the dataset. This is a crucial point: on the one hand, we need many points for constructing a ML model as we don't want to just interpolate our data; on the other hand, if we have many, we might lose accuracy for certain values (one single model cannot predict every single case). The dataset has to be representative from the population we want to cover: huge bias in existing datasets, given that they only account for those materials arbitrarily decided to be representative. It is possible to examine the training of the model with the number of samples, which will give an idea of the number of points to be used.

For example, if you want to predict ionic diffusivities, MP has much more Li, Na based compounds than of any other kind for SSE (this will bias your study).

The second item refers to examining the evolution of the training of the model. There are to possible misperforms: undertraining (your model is not sufficiently complex to reproduce the patterns underlying your data) and overfitting (your model does not generalize well, it interpolates your training data). The first is easily to detect (the accuracy of the model is very bad) and to address (just us a model a bit more complex). The second one is much trickier, as the accuracy is already high.

2.1 Supervised/semisupervised algorithms

For this type of models we can actually check the correcteness of the training.

In K-fold validation we divide the data into k sets and, iterating k times, we used k-1 sets together to train de model and the reamining one to compute the discrepancy between predicition and label.

Consider than if we are classifying it is whether true or false, while for regression we have to define some measure of distance (for instance, if we predict 1.3 and the ground truth is 1.7, how big is the discrepancy?).

The most useful, thus also the most studied, ML task is the regression: we predict one property for a given material. This is the problem in which we are going to focus within this workshop; concretely, with neural networks.

2.2 Regression models

The most common quality metrics (known as loss-functions) for regression problems are the mean squared error (MSE, quadratic distance), the mean absolute error (MAE, euclidean distance) or the mean absolute percentage error (MAPE), with general formula:

$$\begin{aligned}
L_{MSE} &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \\
L_{MAE} &= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \\
L_{MAPE} &= \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y_i}}{y_i}\right|
\end{aligned} \tag{1}$$

Please note that you could also define the loss-function that better fits your problem using any other type of distance between ground truth and predicted values.

2.3 Neural networks

Neural networks are modeled after the human brain, composed of different layers of interconnected nodes (neurons) that work together. Each neuron takes the output of some other neuron and weights it with a specific function (the activation function).

The input layer receives the raw data, subsequent hidden layers extract increasingly complex features, while the output layer produces the final prediction.

During training, the weights of the neurons are adjusted to minimize the predefined loss function (in order to perform better). This process is called backpropagation, and it is possible since neural networks are differentiable.

Canonical activation functions are the rectified linear unit (ReLU) or sigmoid functions, with general formula:

$$\begin{aligned} \sigma_{ReLU}(x) &= \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \\ \sigma_{sigmoid}(x) &= \frac{1}{1+e^{-x}} \end{aligned} \tag{2}$$

For the optimization of the NN it is used backpropagation, which is basically the chain rule for obtaining the dependency of the loss function with respect to every weight function. Then, we want to update weights of neurons following the direction of decrease of the loss function. Many optimizers are currently, defined, although the most used by far is the Adam optimizer (whose name is derived from adaptive moment estimation, and is an extended version of stochastic gradient descent, combining the first and second moments to adapt the learning rate for each weight of the neural network). Its mathematical shape is:

$$\begin{aligned} \omega_i^{(t+1)} &= \omega_i^{(t)} - \alpha \frac{\hat{m}_{\omega_i}}{\sqrt{\hat{v}_{\omega_i}} + \epsilon} \\ \hat{m}_{\omega_i} &= \frac{m_{\omega_i}^{(t+1)}}{1-\beta_1^t} \\ \hat{v}_{\omega_i} &= \frac{v_{\omega_i}^{(t+1)}}{1-\beta_2^t} \\ m_{\omega_i}^{(t+1)} &= \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_{\omega_i} L^{(t)} \\ v_{\omega_i}^{(t+1)} &= \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_{\omega_i} L^{(t)})^2 \end{aligned} \tag{3}$$

where ω_i are the weights of the neurons in layer i, α the learning rate, β_1, β_2 the decay rates and ϵ a small positive constant to prevent zero-divisions.

Let's mathematically define the NN. Imagine that each of our samples of data consists of N points, and we want to predict M output variables simultaneously (most of the cases, we are predciting just one value, therefore $M = 1$). Then, the we can define a K-layers CNN as:

$$f\colon \mathbb{R}^N \to \mathbb{R}^M \tag{4}$$
$$\mathbf{x} \mapsto f(\mathbf{x}) = f_K * f_{K-1} * \dots * f_1(\mathbf{x}) \tag{5}$$

where $*$ express the convolution operation and:

$$f_i\colon \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}} \tag{6}$$
$$\mathbf{x} \mapsto f_i(\mathbf{x}) = \sigma(\omega_i \mathbf{x} + \gamma) \tag{7}$$

where γ are the biases of the neurons in layer i.

Clearly, a NN with no hidden layers and activation funtion the identity is a linear regression, with $f(\mathbf{x}) = \omega_i \mathbf{x} + \gamma$.

2.3.1 Types of NN

The universal approximation theorem implies that any problem can be modeled by a neural networks with specific weights as accurately as we want.

There are various types of neural networks, each designed to solve specific problems. Some of the commonly used types of neural networks are:

- Feedforward NN (FNN): also known as a multilayer perceptron (MLP), this is the simplest form of a neural network where information flows in only one direction, from input to output, through multiple layers of neurons.
- Convolutional NN (CNN): originally designed to process images, they are specially useful for data which is expected to be highly correlated with close elements.
- Recurrent NN (RNN): designed to process sequential data, mantaining data over time, such as time-series data or natural language, allowing nodes to affect themselves (creating cycles inside the network).
- ...

As you may note, any of these models require extracting a vector of input data. This is, for predicting something on our material we need to extract some descriptors. This is not only tedious, but also highly biased as it depends on our prior knowledge (which characteristics of a material correlate the most with absorption?). Although we are still using this physically-informed approach as a complementary tool, there are new approaches for overcoming this issue.

Nowadays, its becoming increasingly interesting a relatively new approach: NN fed by graph-structured data. This is specially interesting for materials science, as we can easily represent a molecule or a crystal with this approach.

This leads us to Graph CNN (GCNN). For each node in a graph, we define some descriptors such as atomic radius, atomic mass, electron affinity, etc. (the feature vector) and some distance between nodes (edges). Then, a message passing scheme is used to aggregate information from neighboring nodes and update the feature vector of each node. Concretely, the feature vectors of neighboring nodes are combined using a trainable weight matrix, which is shared across all nodes in the graph. The resulting vector is then passed through an activation function to obtain a new feature vector for the node.

This operation, which is performed recursively across multiple layers, with each layer learning increasingly abstract representations of the graph, gives each time a new graph, identical to the previous one but with new, updated feature vectors for each atom.

The output of the final layer can be used for various tasks, such as node classification. However, if we apply a pooling operation (such as addition) of all feature vectors in the final graph, we get a single vector describing it. With a linear regression we directly obtain the regression values (or values).

Concepts:

- Transfer learning: a general model is re-trained for some specifc task.
- Active learning: new data is generated for such cases in which the model is seen to misperform.

3 Applications: graph neural networks

It is worth mentioning that one powerfull branch consits on the development of new materials for doing more efficient DL models: neuromorphic computing. Nevertheless, one of the primary applications of ML in materials science is in the development of new materials. By analyzing large datasets of material properties, ML algorithms can identify patterns and make predictions about the properties of new materials. This can help researchers to design and develop new materials with specific properties, such as improved strength or durability, for a wide range of applications.

Another application of ML in materials science is in the prediction of material properties. ML algorithms can be trained to predict various material properties, such as thermal conductivity, electrical conductivity, and mechanical properties, based on their chemical composition and other parameters. This can help researchers to identify materials with desirable properties for specific applications, such as lightweight and durable materials for aerospace applications.

ML can also be used in materials discovery by analyzing databases of materials properties and predicting properties of new materials that may not have been synthesized yet. This can be a time and cost-effective way to narrow down which materials to experimentally synthesize.

However, codifying a crystal into a mathematical structure might be difficult at first, as ML algorithms typically need an input vector. Mainly, two approaches are followed in literature: represent it all in only vector (SMILES, for example) or use a graph representation. While for the former case can simply use previously developed models, for the latter a new type of algorithms have to be constructed (graph neural networks).

Although both have their advantages and limitations, graphs have not only shown very powerful results, but also seem more easy to understand (as they are the structure in which we actually tend to image crystals/molecules), thus we are discussing here machine learning application to graph-like structures.

Graphs are mathematical structures that consist of nodes (each atom) and edges (force between two atoms) that describe a given crystal. Graphs are used to model complex relationships and dependencies within materials, what makes them particularly useful as we do not have to handfully extract descriptors. In other words, we can leverage the power of GNNs to learn patterns and make predictions based on the relationships between the nodes. GNNs excel at capturing local and global dependencies, allowing them to effectively reason and propagate information throughout the graph.

Now, let's discuss the implementation of graphs in GNNs. At a high level, the implementation involves two main components: node embeddings and graph convolutional layers.

- Node embeddings: each node in the graph is associated with a feature vector called a node embedding. Node embeddings encode the information about the node and its local neighborhood. In these applications, we start with all the data we can find in a textbook about each atom.
- Graph convolutional layers: graph convolutional layers are the key building blocks of GNNs. These layers perform message passing between nodes to update their embeddings based on the graph structure. The message passing operation typically involves aggregating information from the node's neighbors and combining it with the node's own embedding. This process allows nodes to exchange information and learn from their local and global contexts. Graph convolutional layers can be stacked to capture increasingly complex patterns and dependencies in the graph.

In addition to node embeddings and graph convolutional layers, GNNs often incorporate additional components such as pooling layers, attention mechanisms, and skip connections to enhance their expressive power and improve performance on specific tasks.

So, to train a GNN, we typically use a supervised learning approach. We define a task-specific loss function, such as mean squared error for regression, and optimize it using backpropagation and gradient descent. During training, the GNN learns to update node embeddings and adjust the parameters of the graph convolutional layers to minimize the loss function.

Through node embeddings and graph convolutional layers, GNNs effectively model relationships and dependencies in the graph, enabling them to make accurate predictions on a variety of tasks.

4 Challenges and new perspectives

While there are many potential benefits to using ML in materials science, there are also several challenges that must be overcome. One of the biggest challenges is obtaining high-quality data. Data can be difficult to obtain in materials science, as it often involves complex and expensive experiments. Additionally, data may be incomplete or inconsistent, which can make it difficult for ML algorithms to accurately predict material properties.

Another challenge is the lack of interpretability of some ML models. Complex models like deep neural networks may provide highly accurate predictions, but their internal workings can be difficult to understand, making it challenging to extract insight from the predictions or even detect biases.

Attending to GNN, there are several challenges that researchers commonly face:

- Graph representation: materials can have complex structures, and finding the right level of granularity for nodes and edges is crucial. Selecting relevant features to represent nodes and edges, as well as defining the connectivity between them, requires domain expertise and careful consideration. For forces we just focus in distances, or should we include some electrostatics as well?
- Graph size and scalability: materials science often deals with large and complex graphs, such as crystal structures or molecular graphs. Handling such large graphs efficiently and scaling GNN models to handle them becomes challenging. Techniques such as graph sampling, hierarchical approaches, and graph pooling mechanisms can be used to address these challenges and improve scalability.
- Limited and noisy data: obtaining labeled data in materials science can be costly and time-consuming. Moreover, the data might be limited in quantity, and the labels can be noisy or incomplete. Dealing with limited and noisy data requires techniques such as data augmentation, transfer learning, and semi-supervised or unsupervised learning methods to make the most of available information.
- Generalization to new materials: GNN models need to generalize well to new, unseen materials. The challenge lies in training models that capture the underlying physics in a way that allows them to transfer knowledge effectively to new materials. Techniques like domain adaptation and transfer learning can help improve generalization to unseen materials.
- Interpretable predictions: developing techniques to interpret and explain the predictions made by GNNs in materials science applications is an ongoing challenge, closely related with understanding their optimal graph representation.
- Integration of domain knowledge: physics-informed machine learning is an indispensable tool. However, effectively integrating domain knowledge into GNN architectures and training procedures requires careful consideration and expertise.

Addressing these challenges requires a combination of expertise in materials science, graph theory, and machine learning. Researchers in the field are actively working on developing novel architectures, algorithms, and techniques to overcome these challenges and advance the applications of GNNs in materials science.

Another topic that has been gaining increasing importance is the bias in current databases: as their consist on previously explored materials for some desired target, moving to new candidates is somehow challenging. In this regard, a new approach appeared recently, generative algorithms. For those how might not know about that, these algorithms are the core of some very wellknown tools such as ChatGPT or Dall-E.

The main idea here is to show a model some existing materials, and explain to it that target you are train maximize. From that, the algorithm is able to create new materials which although share some similitudes with the initial data, might present much better properties. This is known as the inverse problem (from properties, predicting the materials).

Other topics refer as well to designing chemical roots for the synthesis of materials.

5 Conclusions

Thus, the integration of ML into materials science has the potential to greatly accelerate the development of new materials with desirable properties. However, challenges remain in obtaining high-quality data and interpreting the results of ML models. As these challenges are addressed, ML will become an increasingly valuable tool for materials scientists, opening up new avenues for discovery and innovation.