Improving the general optimization strategy with a data-driven framework

Martin van der Schelling (m.p.vanderschelling@tudelft.nl)[1] and Miguel A. Bessa (miguel_bessa@brown.edu)[2]

[1]Doctoral Researcher at Delft University of Technology, The Netherlands
[2]Associate Professor at Brown University, USA

Fundamentals of AI — August 23, 2022

Each group has to deliver **one PDF report** and a **Git repository**. The code should be easy to read, properly commented and it should be possible to replicate the results of the report.

Introduction

Materials science is a substantial field of science as it provides the building blocks for modern engineering applications. Conventional materials design is based on a trial-and-error procedure, as new materials are manufactured through grid-search and tested accordingly to observe their properties.

However, these methods become impractical when the design space increases massively due to the overwhelming combination of different materials. To overcome this challenge, predictive physics-based models and molecular dynamics (MD) simulation have been developed. But despite the ever-increasing potential of computational capabilities, these methods alone are still impractical for extensive design spaces.

In the past decade, machine learning techniques have become popular among the field of materials design. These techniques are able to discover patterns in input data, reducing the design space. A new generation of data-driven material design could be initiated by feeding adequate data from predictive models into machine learned models. Once a model is trained, evaluating a material property using the model is significantly faster than measuring the property in an experiment or computing it in a simulation [1, 2].

F3DASM framework

The framework for data-driven design and analysis of structures and materials (`F3DASM`) is an attempt to develop a systematic approach of inverting the material design process [3]. The framework integrates the following fields:

- Design of experiments (DoE), where input variables describing the microstructure, properties and external conditions of the system to be evaluated are sampled and where the search space is determined.
- Computational analysis, where simulation software is used to create a material response database.
- Machine learning and optimization, where we either train a surrogate model to fit our experimental findings or iteratively improve the model to obtain a new design.

In this way, we motivate the desire to improve the material design process by data acquired from computational analysis models.

The package is written in pure Python and still in the early days of development. Therefore relevant simulation software is not yet implemented. But you are given the chance to work with this piece of software early!

Optimization

During this project we focus on the optimization part of the F3DASM framework. During almost any non-polynomial time machine learning problems, you will encounter some form of an optimization case.

Generally you are trying to find the best parameter of some objective. This objective function $f(\vec{x})$ depends on d number of input parameters represented as a vector $\vec{x} = (x_0, ..., x_d)$. These parameters -with some optional equality or inequality constraints- will span the feasible solution space X. Each point in the solution space could, for instance, represent a micro-structure arrangement or material

Evaluating a sample of the objective function will result in an output vector $\vec{y}$. In analogy to the micro-structure input, this can represent some mechanical properties. For this project, we only consider single-objective optimization, i.e. where $\vec{y}$ is just a scalar. Generally we don't know the underlying objective function. We can only get information by sampling from it.

Trying all possibilities to find the optimum is not really feasible, as it requires a lot of (computational) resources and time. Instead, we try to iteratively improve the current solution with an optimization algorithm.

For each iteration t, the current solution $\vec{x}_t$ is altered to acquire a new solution $\vec{x}_{t+1}$. Ultimately, we want to find an optimal set of parameters $\vec{x}^*$ that will minimize or maximize the objective function $f(\vec{x})$. We can express a minimization optimization problem with the following formulation:

$$f(\vec{x}^*) \leq f(\vec{x}) \; \forall \; \vec{x} \in X \tag{1}$$

A minimization in $f(\vec{x})$ is the same as a maximization in $-f(\vec{x})$. For the sake of consistency, we will view the optimization as a minimization problem for f.

Difficulty with problem-specifics

Unfortunately, designing one algorithm to rule them all is not really an option. According to the famous paper of Wolpert et al. [4], algorithms tend to exploit certain problem-specific characteristics. If this particular feature is not present in the optimization problem, chances are that your performance will be worse than random search, i.e. random sampling. To come prepared for any kind of optimization problem, we need a diverse set of algorithms to mix and match.

Conventional machine learning starts with pre-determined optimization conditions, usually hand-engineered by human experts [5]. However this has several limitations, such as poor generalization to other tasks, severe performance sensitivity and requiring expert knowledge on machine learning to implement the right model.

Meta-learning attempts to learn the optimization conditions itself on a distribution of learning tasks, adding a hierarchical layer to the optimization process [6].

During my masters thesis, I have explored the possibilities of a implementing a general optimizer that can adapt to problem-specific features of the input data. The thesis proposed a data-driven framework for general optimization problems to adapt the algorithm during optimization.

With an adaptive data-driven optimization strategy, we eliminate the need to hand-design a machine learning model. By integrating this idea into the `F3DASM` framework, the proposed software package does not require any machine learning architectural choices beforehand. This will increase accessibility of the framework to pure experimentalists.

Aim of the project

The aim of the project is to design a general optimization strategy with machine learning that competes with the conventional hand-engineered optimizers. You will investigate how different optimization algorithms behave on different problem-specific characteristics.

You will use the F3DASM framework to set-up your search space, sampling, objective functions and optimizers. You will assess the optimizers on various benchmarks, come up with fair performance metrics and

At the end of the project you will gain:

1. Understanding the key challenges of selecting appropriate machine learning models for applications.
2. Knowledge on different optimizers and why they do or do not work.

Preparations

- In order to assist you effectively during the project, please install the following:
  - Create a GitHub account
  - Install GitHub Desktop for Windows/Mac or Linux Ubuntu.
  - Follow the F3DASM installation instructions from the documentation found here.
  - Create a new GitHub repository; this will be the place where you store your code.

  - Add me as a collaborator (my username is `mpvanderschelling`).

- Please <u>read</u> the following material:
  - To understand the problem and ambiguity of optimization: The literature review of my MSc thesis [7], which can be downloaded here.[1]
  - To understand the idea of the framework: Miguel's paper on the F3DASM framework [3].
  - To see an implementation of machine learning and optimization into materials design: The 'fragile becomes supercompressible' paper of Miguel [8]. Also check out this short video about it here!

Getting started

To get familiar with the `F3DASM` package, try to replicate the following experiments:

1. We start of by creating and sampling from a design space:
   1.1. Create a 2D dimensional continuous design space with a single-objective output. Set the boundaries for all of the parameters from -1.0 to 1.0.
   1.2. Sample 50 points from this design space with the `RandomUniformSampling` sampler. Plot these samples for visualization.
   1.3. Repeat 2. for the `SobolSequenceSampling` and `LatinHyperCubeSampling`. What is the difference?
   1.4. Sampling is important for initialization of optimization algorithms. Can you explain why ?
2. Next we instantiate several objective functions:
   2.1. Choose a function that is compatible with a 2D input space from <u>each</u> of the following categories found in the objective function list at the documentation:
      - Convex functions
      - Separable function
      - Differentiable functions
      - Multimodal functions

   2.2. Plot a 3D visualization of the loss-landscape. Use the same boundary values from 1.1.
   2.3. Plot a top-down of the loss-landscapes for each of the functions. Indicate the global minimum on the plots.
   2.4. Create the same heatmap of the loss-landscape but with the sampling points from one of the samplers used in exercise 1. Tip: you may want to use the `plot_data()` function!
3. Now we are ready to optimize our objective functions:
   3.1. Select <u>two</u> of the optimizers from the optimizer list in the documentation and shorty explain for each of them:
      - How the update step of the optimizer works.
      - What the prerequisites of the optimizer are. In other words; are there any constraints on how the optimizer can be used?
      - When this optimizer is effective and when it is failing.

      Support your answers with scientific references.
   3.2. Create an instance for each of the optimizers picked in 3.1 with the default hyper-parameters and the data from one of the samplers. Set the seed so that the experiment can be replicated later.
   3.3. Run the optimization with the `run_optimization()` function for 500 iterations for each of the chosen functions.
   3.4. Create a plot showing the objective function value on the vertical axes and the iteration number on the horizontal axes for each of the optimization trials.

[1] You can skip the part about bio-based composites, although it could help you understand the relevance of machine learning for materials design.

4. The initial conditions of the optimization process can heavily influence the performance. Therefore it is important to redo the experiment with different initial conditions. These are called realizations.
    - 4.1. Run the same optimization of 3.3 with the `run_multiple_realizations()` function for 500 iterations with 10 different realizations.
    - 4.2. Create a plot showing the mean objective function value on the vertical axes and the iteration number on the horizontal axes for each of the different optimizer trials.
    - 4.3. Explain the results of your optimization:
        - Which optimizer is better on each of the objective functions and why do you think this is?
        - Do the results comply with your findings in scientific literature?
5. A big thing in improving the optimizer is to tune the hyper-parameters. We have been using the default settings but maybe we can do better by changing them!
    - 5.1. Choose one of the objective functions of the results created in 4.2 and select the worst-performing optimizer.
    - 5.2. Choose one of the hyper-parameters of this optimizer and explain what it controls.
    - 5.3. Create different instances of the selected optimizer with a different value of the selected hyperparameter
    - 5.4. Run the same optimization process as in 4.2 for the different hyperparameter settings.
    - 5.5. Plot your results and try to improve the optimizer!

Creating a performance metric

stub

Create your own a smart optimization strategy

stub

References

[1] Sanket Kadulkar, Zachary M. Sherman, Venkat Ganesan, and Thomas M. Truskett. Machine Learning-Assisted Design of Material Properties. *Annual Review of Chemical and Biomolecular Engineering*, 13(1), mar 2022.

[2] Kai Guo, Zhenze Yang, Chi-Hua Yu, and Markus J. Buehler. Artificial intelligence and machine learning in design of mechanical materials. *Materials Horizons*, 8(4):1153–1172, 2021.

[3] M. A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, W. Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320(April):633–667, jun 2017.

[4] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[5] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[6] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[7] Martin van der Schelling. A data-driven heuristic decision strategy for data-scarce optimization: with an application towards bio-based composites. mathesis, Delft University of Technology, March 2021.

[8] Miguel A. Bessa, Piotr Glowacki, and Michael Houlder. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. *Advanced Materials*, 31(48):1904845, 2019.