Learning to optimize with a data-driven framework

Martin van der Schelling (m.p.vanderschelling@tudelft.nl)[1], Deepesh Toshniwal (D.Toshniwal@tudelft.nl)[2], and Miguel A. Bessa (miguel_bessa@brown.edu)[3]

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

IFEEMCS520100 Fundamentals of Artificial Intelligence — September 5, 2022

Objective: Use machine learning to choose optimizers for an unseen problem – "learning to optimize".
General Instructions: 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

Conventional materials design is based on experimental trial-and-error. However, just like solving an exam by trial-and-error requires many attempts, this is an impractical strategy because an overwhelming number of combinations become possible as the design space increases. One way to address this issue is to create much more data by using predictive physics-based models[1], but simulating each design can be slow because these computer analysis need to be accurate. This remains true even considering the ever-increasing computational resources.

What if we had a more "intelligent" approach to design? In the past decade, machine learning techniques have soared to prominence in many applied fields of science and engineering. These techniques are able to discover patterns in input data, reducing the design space and informing the optimization process. The new generation of data-driven material design is here where data from predictive models and experiments can be used by machine learned models to discover new things. 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.
- Data generation, usually by computational analysis, where a material response database is created.
- 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 assignment the data generation part is very simple, as we will not be using any numerical method[2]. Instead, we will focus on the optimization part of the framework. The package is written in Python and it is still in the early days of development.

Optimization

Optimization is a critical part of the data-driven process. The goal is to find the best parameters for some objective defined by the objective function $f(\mathbf{x})$ that depends on an input vector $\mathbf{x} = (x_1, ..., x_d)$ with d variables and, in some cases, subjected to some equality or inequality constraints that define the feasible solution space $\mathcal{X}$. Each point in the solution space could, for instance, represent a micro-structure arrangement of a material.

[1]Such as finite element analyses, molecular dynamics, density functional theory, etc.
[2]You will not need to setup nor run computer simulations to create new data.

Evaluating a sample of the objective function will result in an output vector $\mathbf{y}$. 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 $\mathbf{y}$ is just a scalar. Note that 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 $\mathbf{x}_t$ is altered to acquire a new solution $\mathbf{x}_{t+1}$. Ultimately, we want to find an optimal set of values $\mathbf{x}^*$ that will minimize or maximize the objective function $f(\mathbf{x})$. We can express a minimization optimization problem with the following formulation:

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

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

Learning to optimize

When solving a new optimization problem, we do not know what is the best optimization algorithm and its parameters that would quickly get us to the optimum. Unfortunately, the "one optimizer to rule them all" does not exist. According to the famous paper of Wolpert et al. (4), algorithms tend to exploit certain problem-specific characteristics, and if this particular feature is not present in the optimization problem then the performance of the optimizer will be worse than random search, i.e. random sampling (random trial-and-error!).

However, what if we could train a machine learning model on many different optimization problems considering many different optimizers? The machine learning algorithm would start to learn when an optimizer should be used or not. If it is trained properly, then the next time we consider a new problem this algorithm could make an informed decision on which optimizer to use at a given iteration! This is a new field of research called "learning to optimize".

Adequate training so that a machine learning algorithm can "learn to optimize" is challenging because being prepared for any kind of optimization problem requires a diverse set of algorithms to mix and match and a diverse set of optimization problems to learn from.

In the masters thesis of van der Schelling (5), different possibilities of a implementing a general optimizer that can adapt to problem-specific features of the input data have been investigated. 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, and come up with fair performance metrics.

At the end of the project you will gain:

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

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 Martin van der Schelling as a collaborator (Github username is `mpvanderschelling`).

In the end, you should have two git repositories: one containing the code from `F3DASM` and a newly created one for your group.

- Please read the following material:
  - To understand the problem and ambiguity of optimization, read the literature review of this thesis (5), which can be downloaded here.[3]
  - Read this blogpost about the famous 'learning to optimize' paper.
  - *Optional*: To understand the idea of the framework: read the original article on the `F3DASM` framework (3).
  - *Optional*: To see an implementation of machine learning and optimization into materials design, read the 'fragile becomes supercompressible' article (6). You can also check out this short video about it here!

Questions to be answered

Getting started

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

1. We start off 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. Set the `seed` of the sampler to 42. 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. Explain the characteristics from each of the different categories of objective functions implemented in the `F3DASM` framework found in the objective function list at the documentation:
      - Convex functions
      - Separable functions
      - Differentiable functions
      - Multimodal functions
   2.2. Create 2D instances of the `DixonPrice`, `Ackley`, `StyblinskiTank` and `Branin` function. Make sure the `seed` is set to 42.
   2.3. Plot a 3D visualization of the loss-landscape. Use the same boundary values from 1.1.
   2.4. Create a contour plot (a top-down view of the loss-landscapes with its contours) for each of the functions. Indicate the global minimum on the plots.
   2.5. 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] You can skip the part about bio-based composites, although it could help you understand the relevance of machine learning for materials design.

3.1. Create the `Adam`, `CMAES` and `PSO` optimizers from the optimizer list in the documentation and shortly explain for each of them:

- How does the update step of the optimizer work?
- What are the prerequisites of the optimizer? In other words; are there any constraints on how the optimizer can be used?
- When is this optimizer effective and when is it not?

Support your answers with scientific references.

3.2. Create an instance for each of the optimizers with the default hyper-parameters, the data from one of the samplers and `seed` set to 42.

3.3. Run the optimization with the `run_optimization()` function considering 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.

Multiple realizations

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?

Hyper-parameters

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 hyper-parameter

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

6. Comparing the performance of two different optimizers on a single instance of a benchmark function is relatively straightforward (we can i.e. compare the y^*-values). However, we are interested in the overall performance on many different optimization problems. This might cause some problems, since the absolute value of the global minimum might differ.

6.1. Create a diverse set of 100 optimization problems from the benchmark problems found in the `F3DASM` library. Explain why you think that this set represents many different problem-specific characteristics [4].

6.2. Select 5 different optimizers from the `F3DASM` library and use the default hyperparameters. Optimize each problem from the set in 6.1 for each optimizer for 500 iterations and 10 realizations and store this data [5].

[4] Hint: you might consider problems of different dimensionality.

[5] It might be useful to use the json-library for writing Python-objects to disc.

6.3. Compare the performances of the 5 chosen optimizers on the entire set of optimization problems and justify the metric used. You can use a metric from a scientific paper or design your own.

Create your own smart optimization strategy

7. Now it is your turn to design a meta-optimizer!

   7.1. Create a new optimizer by inheriting from the `Optimizer` base-class[6].

   - You can use the 5 different update steps or a combination of them from the optimizers chosen in 6.2.
   - You can only use the information gained by sampling the optimization problem to make an informative decision. For instance, using the name or `global_optimum` attribute of the benchmark problem to train your meta-optimizer is not allowed.

   7.2. Split the 100 optimization problems from 6.1 into a training set of 80 and a test set of 20 problems [7].

   7.3. Train your optimizer on the training set and measure the performance on the test set. Compare the performance with the individual 5 optimizers chosen in 6.2.

   7.4. Reflect on the performance of your optimizer. When is it effective and when not? How does the performance change with a different training set?

Report

Each group has to deliver **one PDF report** and a **Git repository**.

- Use Jupyter notebooks for the Python code. If you are using external libraries, mention them and the used version in the Jupyter notebook.
- The code should be easy to read, properly commented and it should be possible to replicate the results of the report.

References

[1] S. Kadulkar, Z. M. Sherman, V. Ganesan, and T. M. Truskett, "Machine Learning-Assisted Design of Material Properties," *Annual Review of Chemical and Biomolecular Engineering*, vol. 13, mar 2022.

[2] K. Guo, Z. Yang, C.-H. Yu, and M. J. Buehler, "Artificial intelligence and machine learning in design of mechanical materials," *Materials Horizons*, vol. 8, no. 4, pp. 1153–1172, 2021.

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

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[5] M. 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, Mar. 2021.

[6] M. A. Bessa, P. Glowacki, and M. Houlder, "Bayesian machine learning in metamaterial design: Fragile becomes supercompressible," *Advanced Materials*, vol. 31, no. 48, p. 1904845, 2019.

[6] Learn how to do that here!

[7] Bonus: Use k-fold cross validation to account for a bias in the training/testing set.