Software Engineering Assignment

Name: Paul Häusner

Introduction

> Briefly introduce your research area and topic so the rest of your essay stands on its own. (Maximum 500 words)

In my research, I am combining machine learning and numerical optimization algorithms. In particular, we are interested in augmenting optimization algorithms with learned components typically implemented by neural networks. This field of research is known under the concept of learning to optimize and can be applied to very different kinds of optimization problems. In AutoML, the goal is to automatically optimize typically non-differentiable hyper-parameter search for machine learning models. Another popular research direction is learning to learn where tthe goal is to learn the optimizer to train neural networks themselves. In the project, we focus on convex optimization problems as a class of problems that can be solved globally. The goal of my research is to develop learned optimization algorithms that maintain convergence guarantees.

On the machine learning side of the project, one focus is exploiting the connection of graph neural networks and sparse linear algebra [1]. Linear algebra subroutines are some of the most basic operations underlying many if not all numerical methods in modern systems. Even though they are a fundamental tool, many of the algorithms such as preconditioners, starting guesses, and reordering methods are hand-engineered. In practice, we often need to solve many related problems arising from a specific application. For example, structural partial differential equations are typically solved by discretizing the problem domain and applying the finite elements method which boils down to solving large-scale linear equation systems. By using learning methods we can train heuristics and solvers that are tailored to the underlying problem distribution. This allows us to accelerate the solving process on the specific problem distributions of interest while maintaining worst-case guarantees.

We focus both on optimization problems arising in linear algebra such as solving large-scale linear equation systems described above as well as considering more complex optimization problems arising in scientific applications such as medical image reconstruction and battery age modeling. Typically these problems are very large-scale and ill-posed making it time-consuming to solve the optimization problem while in practical applications these problems are time sensitive making it a natural choice to accelerate as there are typically well-defined data distributions parameterizing the optimization problems.

One of the main challenges of applying learning to optimize algorithms is how to deal with distribution shifts of the underlying problem domain. One potential solution for this is to learn another machine learning model that predicts the future distribution of problems [2]. We can then learn a second model to accelerate the solution methods of the predicted future problem distribution. This is especially of interest in scenarios where the data distribution shifts over time. Typically algorithms focus on improving the average performance of the data distribution. However, it is also possible to train the model to accelerate worst-case or quantiles of the loss functions instead. The particular choice depends on the problem domain. However, providing guarantees for the worst-case performance using learned methods is a challenging area related to performance estimation problems (pep).

Lecture Principles

> Pick two principles/ideas/concepts/techniques from Robert’s lectures.
> Discuss how each relates to your research and topic.

Behavioural software engineering: One problem of augmenting numerical optimization algorithms with machine learning is ensuring the convergence to the minimizer of the function. Even though the true minimizer can be achieved results often only holds in expectation over a data distribution and are thus weaker then classical methods. However, in practical problems it is usually not possible to run the algorithm for sufficiently many steps to achieve convergence in practice. Instead, algorithms are typically stopped early which can even have a positive effect. Nontheless there us a resisance towards giving up the classical giarantees from particioners in the field as it is a common practice in the group of researchers in this area. Even though many classical results require unrealistic settings, such as decreasing noise levels or infinite steps in an optimization prodedure, it is difficult to convince practicioners to use algorithms with different kinds of guarantees even though the performance of the models is often signifiantly better in practice. This is an example of the resistance to change towards AI systems.

Verification and validation: Since we are developing mathematical tools to ensure convergence of the optimization algorithms it becomes easier to verify and validate the learned components of the systems; in order to verify the theoretical results and their implementation we simply check the convergence of the optimization algorithm. This can be achieved by comparing the objective value and iterates with the ones produced by a baseline iterative scheme. This can be acheived using dynamic techniques using the baseline algorithms, which ar etypically available in open source libraries, as test oracles. The goal of the learned optimizers is to accelerate this solution process on a specific distribution of problem instances. This allows us to validate the solution by comparing solving time and number of iterations with the baseline method.

Guest Lecture Principles

> Pick two principles/ideas/concepts/techniques from the guest lectures.
> Discuss how each relates to your research and topic

Requirement engineering - problem space: The problem space of my project is readily outlined in the description of my research: we want to develop learned optimization algorithms with convergence guarantees that converge faster than the baseline methods. However, for each specific project these requirements need to be further adjusted to account for the problem specific requirements. For example, in my first research project [3], a key problem-class specific requirement in the problem space is that the neural networks outputs a lower triangular matrix which is the Cholesky factor of a positive definite matrix. Already defining this requirement requires domain specific knowledge but is a key step to develop the solution methods.

Requirement engineering - solution space: The solution space is the output of the research. In other words, how can we achieve this is one of the main research questions of my research project. Taking up the example from the previous point, in the solution space we use a graph neural network that takes as input a lower triangular matrix and outputs a lower triangular matrix. The diagonal elements are replaced passed through a activatino function that ensures strict positivness of the diagonal elements.

Data Scientist vs Software Engineering

> Read chapters “1. Introduction” and “2. From Models to Systems” of the CMU “Machine Learning in Production” book (https://mlip-cmu.github.io/book/01-introduction.html) and then answer:

> Do you agree on the essential differences between data scientists and software engineers put forward in these chapters? Why or why not?

There are arguments for both for and against the distinction made in the book. However, overall I believe that the distinction presented in the book chapters are too two-dimensional. Both the role of data scientist and software engineer are presented as complete and coherent groups which is not the picture of the real world. The software produced by engineers always needs to fulfill some additional requirements to be useful in practice and, thus, a "pure" software engineer is always useless as they are not able to create a product with value for a company. For example, front-end engineers need to build websites that are accessible with thought about user experience but also need to integrate into the buisness case. Similarly, backend engineers are required to take into account the security of the system and potentially data storage. There are also roles that support these engineers for example designers create a concept of the front end that then gets realized by front-end engineers. In some cases the designer and the engineer can be the same person but not neccessarily. Another examples is software architects which mainly focus on building a coherent stack of technologies the engineers are using. A data scientist can cover a wide range of problems as it is presented in the book: from a full science standpoint, they can only experiment with models in a Jupyter enviornment to a role where they support bringing the final model into the greater picture.

Thus, in summary, creating a product always requires different roles and skills. Only by collaborating between these different roles a sucesfull project can be carried out. However, creating a clear distinction between specific roles can already create imaginary boundaries in a team which makes the development process more complicated.

> Do you think these roles will evolve and specialise further or that “both sides” will need to learn many of the skills of “the other side” and that the roles somehow will merge? Explain your reasoning.

I think we will see more people with interdisciplinary skills that have an understanding of data science. Thus, we will see new roles in software teams emerging that combine existing roles with roles required for data scientists. For example, database experts can support in building a data infrastructure, software architects can help building continous integration pipelines, and test engineers can help build and define ways to automatically test machine learning methods. Ultimalty, the way the ML components are used in the product will dictate which kind of roles are needed in the project. However, there will always be a need for interdiscinplinary skills and creating artificial boundaries between different roles can be harmful in a project and limit its success.

Paper Analysis

> Find two full/long papers from any CAIN conference (links at https://conf.researchr.org/series/cain), download and read each, then for each paper cover: (1) Core idea and their SE importance (2) Relation to your research (3) Integration into a larger AI-intensive project (4) Adaptation of your research

Paper 1: A Combinatorial Approach to Hyperparameter Optimization

The authors of [4] suggest utilizing ideas from software testing to hyper-parameter optimization of machine learning methods. For a given model, they first manually indentify a set of important hyper-parameters and then generate a set of t-tests where t dictates how many of the parameters interact and can be seen as a hyper-parameter of the testing framework that dictates the time required to do the hyper-parameter optimization. The tests themselves are generated using a combinatorial testing tool to fulfill potential constraints. For each of the generated tests, the model is trained and the best performing model chosen based on the performance on a predetermined metric. This allows a systemtic traversal of the hyper-parameter search space that is not dependent on randomness or prior information while maintaining scalable for high-dimensional hyper-parameter settings. The contributions of this can be seen as two-folded: firstly, tools from software engineering - in particular from software testing - are used to improve the optimization of models. Secondly, this methodology can be applied to software engineering where often the cost of tuning a model is a limiting factor.

One research direction in learning to optimize is to predict good settings for an optimization solver, such as a starting guess or the next value to consider, which can be a part of the hyper-parameter space when considering training machine learning models. However, mostly we focus on continous optimization which means that the output of the neural network can be further optimized using gradient based optimization techniques. In contrast, most often no gradient information is available for hyper-parameter tuning and instead one has to use black-box techniques instead. These approaches are more challenging from a theoretical standepoint but many of my research projects can be extended to this setting.

An important component in larger AI systems is the automated testing and tuning of models. Condier for example a project where the dataset might change over time and new features are added by data scientist. Always relying on manual tuning of the hyper-parameters is, as we have seen in this course not a good idea. Instead, we want to rely on some automized pipeline. However, naive solutions typically does not scale well due to the large search space and can cost companies significant amounts of money and time. Utilizing the ideas outlined in this paper, it is possible to systematically try out different hyper-parameters without suffering from the course of high dimensionality by restricting the search space. Further, the automatic test case generation allows restricting the search space a-priori by adding specific constraints.

In my research project we usually apply manual hyper-paramter tuning rather than applying a systematic approach. In particular since many of the design choices in our projects have a theoretical motivation rather than purely optimizing the performance of the model. However, using the combinatorial test generation it would be possible to automatically generate hyper-parameter configurations which fulfill the theortical requirements by translating them into constraints. Further, optimizing some of the hyper-parameters in an automated fashion can benefit the results as it leads to better performing models. While in this line of research it is not required to tune the model indefintly, further hyper-parameter optimization over a small search space can increase the model performance.

Paper 2: Worst-Case Convergence Time of ML Algorithms via Extreme Value Theory

In paper [5], the authors suggest to use extreme value theory which is rooted in statistical theory to predict the worst case probabilistic timing bounds of meachine learning models. Predicting these times can be challenging as it can not be achieved using classical static code analysis tools. Obtaining good estimates is of importance from a software engineering standepoint to ensure the availability of resources and mitigating the increasing climate impact of the models as well as to steer informed buisness decisions. The statistical analysis proposed in this paper can be applied to both predicting the training times of models as well as the inference times of machine learning models. This allows engineers in the process to make informed decisions and provide accurate cost estimates for stackeholds as well as improve the scailability of the method to ensure availabilities to users. While the analysis here focuses on the machine learning models themselves it is possible to directly extend the results to considering the whole system containing many other steps such as data extraction which can be analyzed using more classical code analysis tools.

The approach taken here is similar to automated performance analysis (pep) but takes into account the underlying hardware and considerably more complex optimization of general machine learning algorithms. However, no guarantees about the resulting performance or global convergence can be made. Nontheless, in practice it is often very important to take into account the real-world time of the algorithms. In first-order algorithms I mostly consider in my research project providing convergence based on the number of iterations is is usually sufficient as each iteration has a fixed cost which is usually cheap compared to other algorithms.

Most often AI and machine learning components in larger AI systems are trained using PaaS services as provided by big cloud providers such as amazon. Running a large hyper-parameter search can in these cases be a financial and buisness risk for a company as it is uncertain how long it takes until the process converges. Using the tools presented in this paper it is possible to give an a priori estimate of the required worst-case time to execute the full optimization process. This allows a buisness decision before runing the actual optimization to decide whether it is worth the monetary cost to improve the model performance further. It would also possible here to combine these techniques with the ideas outlined in the first paper and increase or decrease the amount of hyper-parameter test cases based on the predicted time it takes to optimize the model.

Most of my experimental results e.g. in [3] contain the experimental inference and training time of the graph neural network utilized in my research. Including worst case bounds for a specific kind of hardware would make the results significantly stronger and benefit particioners to verify that the introduced methods can outperform existing approaches. However, one pitfall with this is that the theory presented here only focuses on the tails of the distribution while a 95% quantile or the median are of bigger interest for my research projects.

Research Ethics and Synthesis reflection

> To make your own literature search transparent describe exactly how you found and selected the papers you based your question 4 answer on: (1) search and screening process (2) pitfalls

> and mitigations (3) ethical considerations

During the search, priority was given to newer papers of the latest edition of the conference as the field is rapidly evolving. However, in order to avoid too many topics on LLMs, also older version of the conference were considered to cover a broader selection of topics. The final selection was made based on finding papers that best match my research and technical aspects of optimizations processes that can be easily integrated and discussed in the context of my own research.

Language models and other tools were only used once the draft of the assignment was completed in order to ensure originality. After completion, tools were used to improve the writing and style of the essay without changing the contents. All used references for the covered topics are included in the reference section below.

References

[1] NS Moore, EC Cyr, P Ohm, CM Siefert, RS Tuminaro, Graph neural networks and applied linear algebra, SIAM Reviews 2025.
[2] Pascal Van Hentenryck, Optimization Learning, arXiv:2501.03443.
[3] Paul Häusner, Ozan Öktem, Jens Sjölund, Neural incomplete factorization: learning preconditioners for the conjugate gradient method, Transactions of Machine Learning Research, 2024.
[4] Krishna Khadka, Jaganmohan Chandrasekaran, Yu Lei, A Combinatorial Approach to Hyperparameter Optimization, CAIN 2024, April 14–15, 2024, Lisbon, Portugal.
[5] Saeid Tizpaz-Niari and Sriram Sankaranarayanan, Worst-Case Convergence Time of ML Algorithms via Extreme Value Theory, CAIN 2024, April 14–15, 2024, Lisbon, Portugal.