Hyperparameter Optimization

AutoML Assignment 2

Evan Meltz, Giulia Rivetti

1 Theory questions

1. In a N-way k-shot classification setting, a given dataset Y is split into three parts: meta-training, meta-validation and meta-test sets. For each of these sets, we sample a batch of tasks, where each task contains a support set with N different classes and k examples per class. For this reason, this approach is known as n-way k-shot.
   Meta-validation tasks are used for the optimization of the metalearner's hyperparameters. In particular the meta-validation tasks were not seen during the meta-training phase and the model's performance on these tasks is used to adjust and optimize the model's hyperparameters so that it can generalize well on other new, unseen tasks.
   The meta-test set is used during the meta-training phase to evaluate the performance of the model after task-specific updates, so that we can test how well a technique performs at metalearning. In particular, each meta-test task contains only unseen classes that were not in any of the training tasks. For each task, we measure performance on the query set based on knowledge of its support set. In this way, one can assess the generalization performance of the meta-trained model to new, unseen tasks[1].

2. In MAML, the meta-gradient refers to the gradient of the task-specific loss against the initialization parameters. The main idea of MAML is to update the initialization parameters (θ) so that the model can quickly adapt to new tasks with little data. The meta-gradient helps do this as it is used to update the initialization parameters during the meta-training phase. Therefore the meta-gradient will be equal to zero when the task-specific gradient cancels out the effect of the intial parameters. This can be displayed as:

   $$\nabla_\theta L_{\text{task}}(\theta - \alpha \nabla_\theta L_{\text{task}}(\theta)) = 0$$

   You can therefore conclude that when the meta-gradient is zero, the models initialization parameters are at an ideal value for that task and θ does not need to be updated further. When the meta-gradient is zero consistently across several tasks, the model's initialization is now at an optimal value to be able to achieve MAML's main goal of fast adaptation to new tasks with little data.

3. In the case where the new encountered task is distant from the training task distribution we would expect MAML to work better than ProtoNet. ProtoNet is a metric-based meta-learning approach, and as such it has the disadvantage of a decrease in the performance when the new task encountered is distant from the ones on which it trained. Indeed when tasks at meta-test time become more distant from the tasks that were used at meta-train time, metric-learning techniques are unable to absorb new task information into the network weights, with the consequence of a

degradation in performance [1]. MAML, which falls into the class of optimization-based meta-learning, doesn't have this disadvantage and it was actually implemented with the objective of learning a good initialization in order to quickly adapt to new tasks; therefore in those specific conditions it would outperform ProtoNet.

4. MAML and Reptile are both meta-learning algorithms, but as mentioned, differ in their optimization procedures with emphasis on their update model parameters during the meta-training process. MAML is known for its ability to adapt to a wide range of tasks with minimal data. It does this by learning an initialization that facilitates quick adaptation to new tasks. Furthermore, MAML is model-agnostic, which means it can be relatively easily applied to different types of models. For these reasons MAML would likely outperform Reptile in tasks of increased complexity and which would require a more nuanced adaptation of its models' parameters. Due to MAML's explicit gradient-based update procedure, the adjustments will be more fine-grained which will benefit complex tasks and yield higher results when compared to a model like Reptile which has a far more basic update approach. In addition, as mentioned MAML is model-agnostic, thereby allowing it to easily adapt to a variety of model architectures which would result in it also outperforming Reptile in any scenario where the optimal architecture for each task is different. However, while MAML is more complex with its method, this of course comes with the trade-off of being more computationally expensive due to the need for multiple inner loop optimization steps. For this reason, Reptile would likely outperform MAML in any situation where computational efficiency is critical. Due to its relatively simple update process, training times would be far shorter, when compared to MAML. Furthermore, Reptile's update process involves a simple linear interpolation between the initial parameters and the parameters after the inner loop optimization which would be highly advantageous over MAML's in cases where the tasks are simpler and don't require highly adaptive initializations. Lastly, unlike MAML, Reptile does not differentiate through the optimization process, therefore making it more suitable for optimization problems where many update steps are required.[2].

2 Empirical questions

For the following parts, we have performed three runs for each configuration (different MAML's orders and parameters) and then we considered the average values of these runs for result consistency.

1. Figure 1 below shows the average of the training and validation losses for three runs of our algorithm for both first and second order MAML.
   From this graph it is apparent that the training loss for both first and second order MAML, start at a relatively high value but it very quickly decreases to an acceptablely low loss value. Considering the algorithm is adapting to the training data, and the model parameters are undergoing significant adjustments, this is unsurprising. With the validation loss however, we can see the value starts a lot lower and stays there, as well as being a lot more stable compared to the training loss which has constant spikes in its value, likely because of the model encountering challenging instances or outliers in the training set. We took this to mean that the algorithm generalizes well to unseen data.
   Lastly, First Order MAML compared to Second Order MAML had no noticable differences to us. This could be because of the specific characteristics of the dataset or the complexity of the

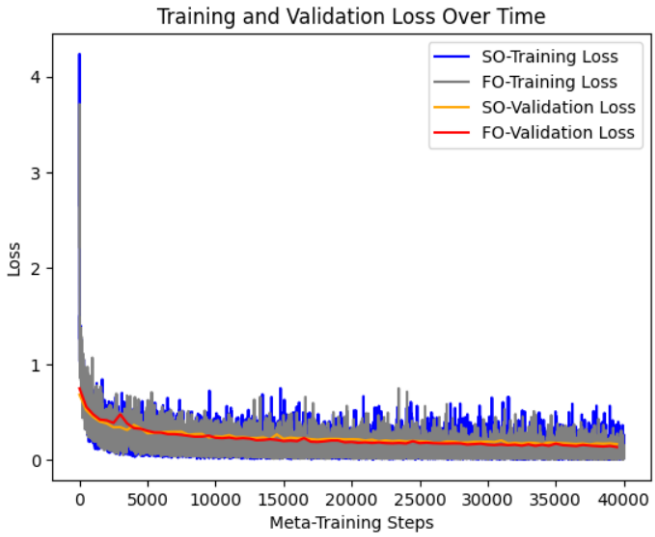


Figure 1: Training and Validation Loss for First and Second-Order MAML

underlying task. Further analysis, like examining the gradients during training, would be required to capture the nuanced distinctions between the two approaches.

2. In Figure 2, we have plotted the test accuracy for both first and second order MAML with three different numbers of inner gradient update steps T: 1, 5, and 10. From the plot, we can see that in general, in all cases, the model is able to achieve good accuracy, since the minimum and maximum whiskers are generally between 0.85 and 1. Moreover, the interquartile range is closer to the upper part of the whiskers' interval. By looking at the plot, we can observe that there isn't a significant difference in the performance when increasing the number of inner gradient update steps from 1 to 5; a small improvement can be seen for T=10, where for first order and especially second order MAML the meta-test accuracy is higher. Therefore we can observe that increasing the number of inner gradient update steps to an appropriate number (such as 10) can have a positive increment in the performance of both first and second order MAML.

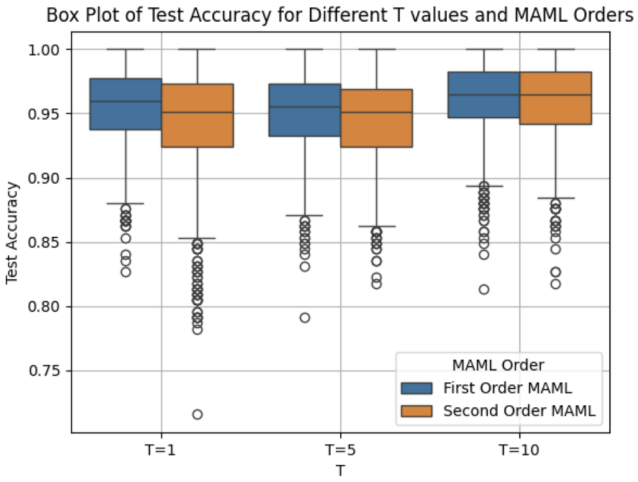
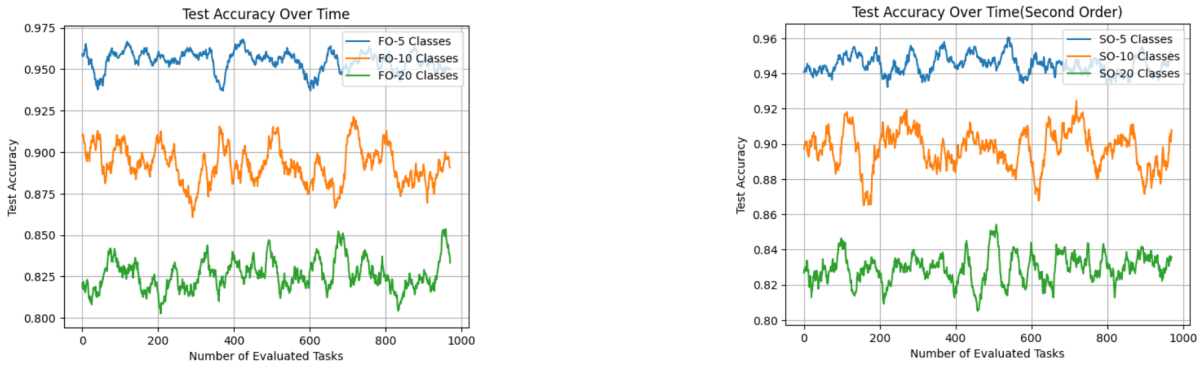


Figure 2: Box-plot showing the accuracy on the test set for first and second order MAML with different values of T (number of inner gradient update steps)

3. Figure3 below shows the effect that increasing the number of classes per task has on the test accuracy for both first and second order MAML. Most noticeably, the increase in the number of classes has a consistent decline on the test accuracy. This is most apparent when seeing the



(a) First Order

(b) Second Order

Figure 3: Test Accuracies with Increased Number of Classes per Task

difference of the accuracy from 5 to 10 is approximately doubled when looking at the difference from 10 to 20. This decrease in test accuracy occurring with an increase in the number of classes per task could suggest that the algorithm faces more of a challenge when generalizing across a larger set of classes. As with empirical question 1, there are basically no changes between first and second order MAML to report.

4. In Figure 4, we have plotted the test accuracy for both first and second order MAML as a function of the number of meta-tasks evaluations and for two different values of the meta-batch size (1 and 4). From the picture we can clearly see how increasing the meta-batch size affects the performance of both first and second order MAML: with the meta-batch size set to 4, the two algorithms are able to achieve higher accuracy on the test set, therefore we can observe that increasing the meta-batch size to an appropriate value (4 in our case) can improve the generalization performance of both first and second order MAML; indeed a larger meta-batch size implies using more task samples to compute the meta-gradient, which can lead to a more accurate estimation of the direction in the parameter space that improves generalization across tasks.

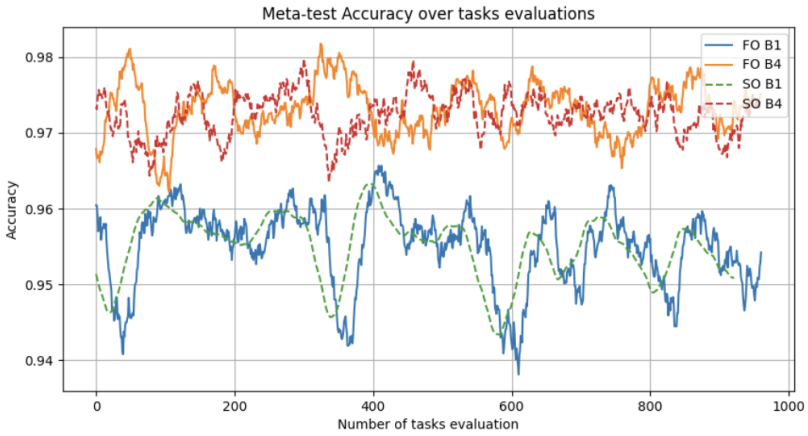


Figure 4: Meta-test accuracy for first and second order MAML with different meta-batch sizes (1 and 4) as a functions of the number of tasks evaluations.

References

[1] Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. *Metalearning*. Springer, 2nd edition, 2022.

[2] Alex Nichol, Joshua Achiam, and John Schulman. Reptile: a scalable meta-learning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.