Deep Meta-Learning

AutoML Assignment 2

Lukas Welzel, Mark Rademaker

1 Theory questions

1.1 We create three sets by splitting the entire dataset of class $y \in Y$ in, for instance, 80% training data, 10% validation data, and 10% test data. These splits all contain different classes, this means that, in our example, 80% of the classes are in the training set, 10% of the classes in the validation set and 10% of the classes in the test set. The training class contains k number of examples in N-way k-shot learning. The key difference here is that we make the splits at class level instead of ordinary data level splits.We use a validation set when we want to determine the optimal hyperparameters for our model. By making predictions on unseen tasks we can validate the model with its current settings used for training. The validation task performance gives an estimate of the loss of the model. We want the model training to be completely independent of the test tasks. The validation score can then be used to change the settings (hyperparameters) of the model such that it performs better on unseen data. We can use the configuration of the model that gives the lowest loss estimation, this is known as hyperparameter optimization. The true loss is than derived from the performance of the optimal model with hyperparameters, according to the validation score, on the test tasks. We use test tasks to evaluate the true performance of our trained model. When we want to make any conclusions about the generalizability of the model we should always use test tasks. To fully evaluate and train a good-performing model train tasks, validation tasks and test tasks are necessary.

1.2 The prediction for $\hat{x}$ for class 1 and 2 are $p_\theta(y = 1 \mid \hat{x}) = 0.73, p_\theta(y = 2 \mid \hat{x}) = 0.27$ respectively.

$$\mathbf{c}_k = \frac{1}{N_C} \sum_{(\hat{x}_i, y_i) \in S_k} g_\theta\left(\hat{x}_i\right), \quad \mathbf{c}_1 = \frac{1}{2}\begin{bmatrix}2\\5\end{bmatrix}, \ \mathbf{c}_2 = \frac{1}{2}\begin{bmatrix}-2\\-5\end{bmatrix} \tag{1}$$

$$-d_{k,\theta} = -\left\|g_\theta(\hat{x}) - \mathbf{c}_k\right\|^2, \quad -d_1 = -8, -d_2 = -9 \tag{2}$$

$$\hat{y}_\theta\left(\hat{x}, D^{tr}_{\mathcal{T}_j}\right) = -\sum_{k=1}^{2} d_{k,\theta}\left(\hat{x}, c_k\right) e_n \quad = \begin{bmatrix}-8\\-9\end{bmatrix} \tag{3}$$

$$p_\theta(y = k \mid \hat{x}) = \frac{e^{-d_{k,\theta}}}{\sum_{k'} e^{-d_{k,\theta}}}, \quad p_\theta(y = 1 \mid \hat{x}) = 0.73, \ p_\theta(y = 2 \mid \hat{x}) = 0.27 \tag{4}$$

1.3 Prototypical network is a meta-learning algorithm because it includes a learning subsystem and exploits meta-knowledge through embedding features in a metric space to find (the closest) distances to prototype representations of classes. Thus, Pnets expect and exploit input similarity.The inner-level learning is determining the centroid of every class, by averaging over the representations of the examples of the support set in the metric-space we learn a representation of a class. Then by adjusting the

parameters of the network based on the loss on the query set, we are able to get better representation outputs from our network, this is the outer-level learning.

1.4 When the loss functions on different tasks cancel each other out for a parameter, this means that updating a parameter will not improve the loss of a task without a decrease in performance for another task, the gradient will then be equal to zero. This means that the initial parameter is optimized. MAML updates its parameters when the sum of second-order gradients of the tasks is not equal to zero for the given support sets.

1.5 We expect the Pnet model to work better than the MAML model. Since both models perform equally well on the training set, and the new example is distant from this training set, we expect that the Pnet model will better in classifying the new task as it uses the distance to measure similarity. The Pnet model can use the fact that the training tasks and the new task are distant as a margin for error for the similarity (distance) estimation. The MAML model needs to have large/many update steps, to adjust it's parameter initialisation and be able to classify the new examples, when the new task is distant from the training tasks. When the initialisation parameters for MAML are too distant to the optimal parameters of the new task the performance of the model will deteriorate. A combination of both, such as ProtoMAML [3] is expected to outperform both, especially for more shots.

1.6 See Snell et al. 2017 [2], following their notation:

$$-\|f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbf{c}_k\|^2 = \underbrace{-f_\phi(\mathbf{x})^\top f_\phi(\mathbf{x})}_{\text{const w.r.t. class k}} + \underbrace{2\mathbf{c}_k^\top f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbf{c}_k^\top \mathbf{c}_k}_{\text{repr. lin. model}} \tag{5}$$

$$\text{for proportionality: } 2\mathbf{c}_k^\top f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbf{c}_k^\top \mathbf{c}_k = \mathbf{w}_k^\top f_{\boldsymbol{\phi}}(\mathbf{x}) + b_k, \text{ where } \mathbf{w}_k = 2\mathbf{c}_k \text{ and } b_k = -\mathbf{c}_k^\top \mathbf{c}_k \tag{6}$$

$$\text{so that } w_{k,n} = 2c_{k,n}, \quad b_{k,n} = -c_{k,n}^2 \tag{7}$$

While this linear model in contrast to a non-linear mapping (e.g. by using a different metric) uses a linear projection onto the w vectors, the resulting (non-linear) error can be approximated in the embedding by the Protonet.

2 Empirical questions

To evaluate the models we do 5 runs per model of 40000 epochs on the omniglot dataset [1]. With these 5 runs we calculate the confidence interval of 95%. After every 500 epochs we validate the model. Then number of validation tasks is set to 1000 of which we take the average score, as a result we will get 80 validation scores and 40000 training scores, which causes a smoother learning curve on the validation set shown in the figures. We run the models with the he 5-way, 1-shot setting.

2.1 Figure 1 shows the training accuracy and training and training losses of ProtoNet, 1st order MAML and 2nd order MAML. Figure 1 shows the loss and accuracy on the validation set of these models. From the figures, we see that all three models learn from the support set as the accuracy of the query set improves for all three models during training. We see that the Protonet model quickly converges to an accuracy close to 100% on the training set in 1. For the MAML models the 2nd order MAML learns faster from the training set than the 1st order MAML, this is expected as the 1st order

MAML might be sufficient to learn, but by skipping the second order gradient we lose information about our parameters.
Figure 1 shows the loss on the validation set, of the three models, the 2nd order MAML will have the lowest loss. The ProtoNet learns quickly but the performance stagnates after 10000 epochs. The 2nd order MAML will outperform the ProtoNet model after 30000 epochs based on the validation loss. The 2nd order MAML model is thus best at predicting the query shots when trained properly.

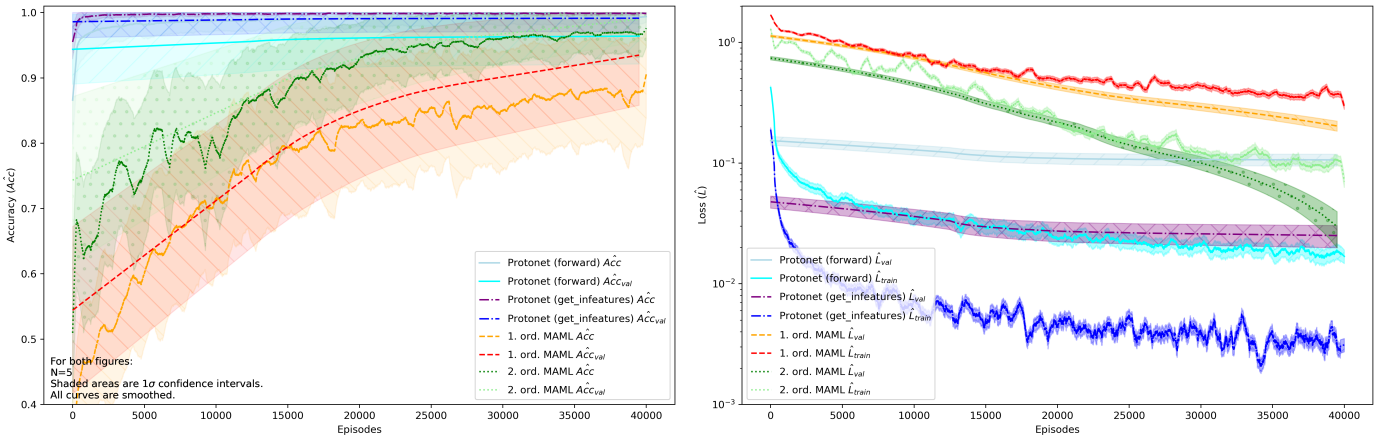

Figure 1: ProtoNet (using `get_infeatures` *and* `forward`), 1st and 2nd order MAML training and validation performance. Note the log-scale on the left figure.

2.2 Multiple inner network steps allow the model to learn the support set better. As a result, the difference between the initial parameters and the trained parameters will likely be bigger when the initial parameters are far from optimal. For 1st order MAML, this will speed up the learning process. As can be seen in figure 2, the left curves show how multiple inner steps will accelerate the learning process. After 5000 epochs the models with multiple inner steps reaches an accuracy that the model with one step only reaches after 40000 epochs. From figure 2 we can conclude that the MAML with 5 and 3 inner steps significantly outperforms the model with 1 step based on the validation loss. A problem may appear when using 2nd-order MAML which requires the model to calculate the gradients of the inner loop as well. When taking multiple inner steps, for instance, 5, we have to calculate the fifth gradient of the inner loss function. This is not only very computationally expensive, but it can also cause instability as gradients can explode or vanish.

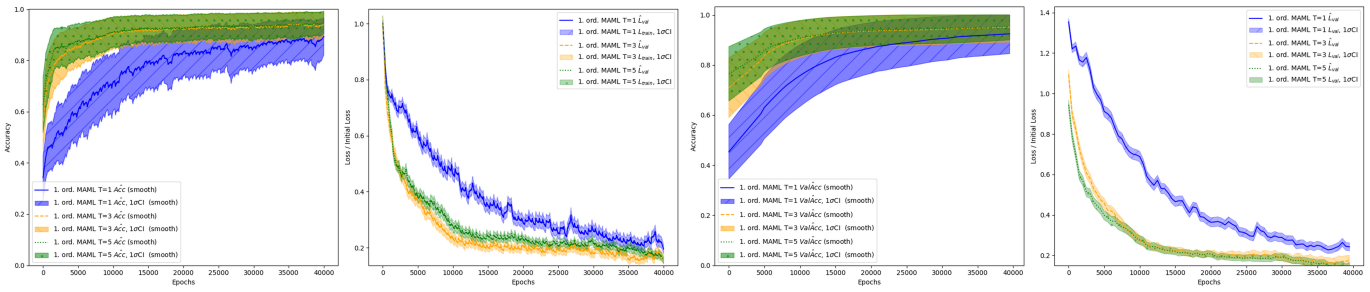

Figure 2: 1st order MAML with 1, 3, and 5 inner gradient step models and their training (left) and validation performance (right).

2.3 We analyze the performance ProtoNet and 1st-order MAML with a different number of classes per task. The ProtoNet model does not show a significant difference in performance when `forward`[1] was used, however using `get_infeatures` differentiates the performance with the number of classes as each task has higher diversity. Nevertheless, this result does imply that a model with more tasks per class does not necessarily generalize better than one with fewer, as long as its data is representative of the actual real-world data. We suspect that the difference between the two Protonet implementations (`forward` vs. `get_infeatures`) is due to the difference in embedding capacity. The MAML model 3 shows a slight under-performance of 10 classes per task compared to 5 and 20 classes per task. However, as the figure also shows an unstable learning process, bad initialization could affect the performance of this model. To make any strong conclusions about the instability shown in 3 we need more runs than 5 to prove that bad initialization is not a factor. Lastly, it is clear that the choice of algorithm has a much larger impact on performance than the number of classes per task.

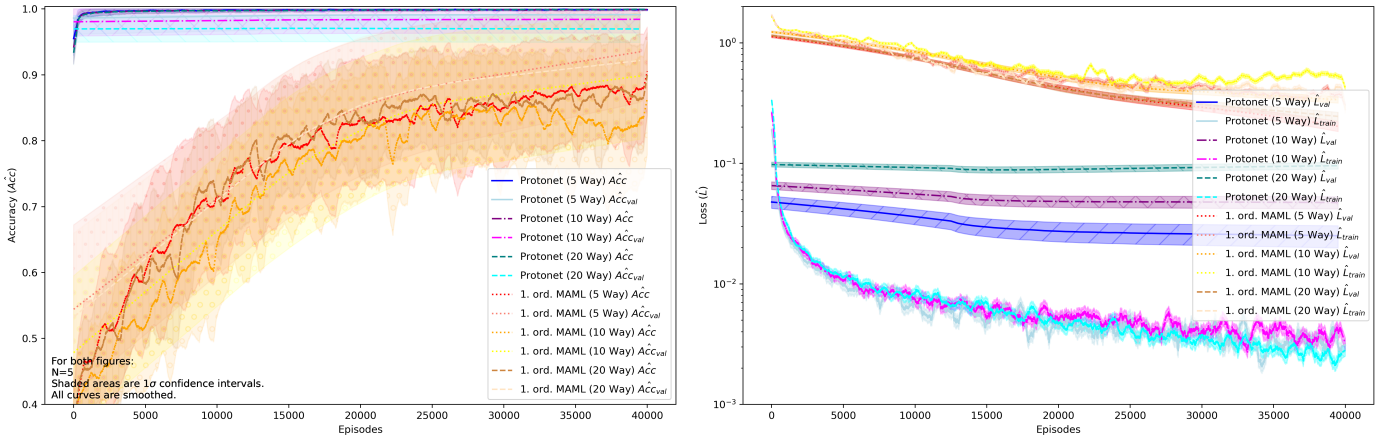

Figure 3: Protonet and MAML performance with 5, 10, and 20 classes per task.

References

[1] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[2] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[3] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[1] Previous report.