Meta-learning for Multi-Task Symbolic Regression

Samuel Kim, [*] **Ileana Rugina** [*]

Abstract

While symbolic regression has traditionally been implemented using genetic algorithms, in recent years deep learning has offered alternative approaches to symbolic regression to augment its capabilities. However, in many scientific and engineering scenarios, gathering observations can be expensive or time-consuming making it difficult to gather large datasets and thus apply standard machine learning algorithms. However, because many experiments stem from the same underlying phenomena, the equations used in a particular field of science or engineering may share common element (i.e. primitive functions or equation structure), providing an opportunity to introduce inductive biases in the form of meta-learning. Here we apply model-agnostic meta-learning (MAML) to the equation learner (EQL) network, a deep learning system for symbolic regression, and solve multi-task regression problems that are generated either arbitrarily or from function distributions. We find that MAML learns a good initialisation that outperforms joint training without fine-tuning in the evaluation phase. We analyse both the joint training baseline and MAML in out-of-distribution setting. Our code is available at https://github.com/samuelkim314/6.883-Project-MetaEQL.

1. Introduction

Symbolic regression is a type of regression analysis that aims to discover analytic equations to describe a dataset. The resulting equation is interpretable by a human and can reveal insights that are useful for discovery in science and engineering. For example, astronomical observations were used to discover Kepler's laws of planetary motion, which eventually led to the discovery of the laws of motion and laws of gravitation. Additionally, these equations can extrapolate well outside of the regime from which they were formed, making them extraordinarily powerful in science and engineering for prediction and design.

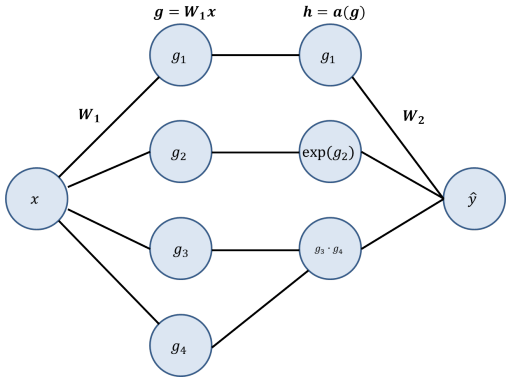

Figure 1. Example of the Equation Learner (EQL) network for symbolic regression using a neural network. Here we show only 3 activation functions and 1 hidden layer for visual simplicity, but the network can include more functions or more hidden layers to fit a broader class of functions.

Since many equations that appear in physical systems often share the same operators and similar forms, we plan to apply meta learning to symbolic regression tasks and leverage the inductive bias of natural laws into symbolic regression systems. Additionally, in a multi-task setting the algorithm can also learn useful priors that apply to different fields, such as different sub-fields of physics.

Traditionally, symbolic regression is solved using genetic programming (GP), in which the equation is encoded using a syntax tree, and the GP algorithm iterates through different tree structures to find the equation that best fits the dataset (Dubčáková, 2011).

A more recent approach is using neural networks in which the primitive functions correspond to the activation functions and the weights after training correspond to coefficients in front of the primitive functions (Martius & Lampert, 2016; Sahoo et al., 2018; Kim et al., 2020). The EQL network (shown in Figure 1) uses primitive functions as the activation units and trains with backprop and a sparsity regularization term to reach an interpretable equation that

[*]Equal contribution . Correspondence to: Samuel Kim <samkim@mit.edu>, Ileana Rugina <irugina@mit.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

describes the dataset. Another system is AI Feynman in which neural networks are used to help find symmetries and simplifications in the data so that it is more amenable to brute-force approaches (Udrescu & Tegmark, 2020; Udrescu et al., 2020). AI Feynman explicitly introduces physical inductive biases through a handful of different techniques such as performing dimensional analysis or explicitly encoding observed symmetries.

In this work, we focus on the EQL network. We use both joint training and MAML to learn a good initialization of the EQL network that can be used in the multi-task setting, which in this case, is regression on a variety of different functions. We train functions drawn from arbitrary distributions as well as from a distribution defined by the damped harmonic oscillator physical system and evaluate the resulting model on in-distribution and out-of-distribution validation functions.

2. Methods

2.1. EQL Network

We briefly review the EQL network architecture here, but more details can be found in (Martius & Lampert, 2016; Sahoo et al., 2018; Kim et al., 2020).

The EQL network is a fully-connected network with input $\mathbf{x}$ where the i^{th} layer of the neural network is described by

$$\begin{aligned}\mathbf{g}_i &= \mathbf{W}_i \mathbf{h}_{i-1} \\ \mathbf{h}_i &= a\left(\mathbf{g}_i\right)\end{aligned}$$

where $\mathbf{W}_i$ is the weight matrix of the i^{th} layer and $\mathbf{h}_0 = \mathbf{x}$ is the input.

The activation function $a(\mathbf{g})$, rather than being the usual choices in neural networks such as ReLU or tanh, may consist of a separate function for each component of $\mathbf{g}$ (such as sine or the square function) and may include functions that take two or more arguments while producing one output (such as the multiplication function):

$$a(\mathbf{g}) = \begin{bmatrix} a_1(g_1) \\ a_2(g_2) \\ \vdots \\ a_{n_h}(g_{n_g-1}, g_{n_g}) \end{bmatrix} \tag{1}$$

A visual example of the EQL network is shown in Figure 1. Note that an additive bias term can be absorbed into $a(\mathbf{g})$ for convenience. These activation functions in (1) are analogous to the primitive functions in symbolic regression. Allowing functions to take more than one argument allows for multiplicative operations inside the network. We also allow for activation functions to be duplicated within each layer (i.e., multiple components in g can use the same activation function).

By stacking multiple layers (i.e. $L \geq 2$), the EQL architecture can fit complex combinations and compositions of a variety of primitive functions.

For these experiments, we use the activation functions in Table 1

$a(g)$	n_g
1	$2m$
g	$4m$
g^2	$4m$
$\sin(2\pi g)$	$2m$
e^g	$2m$
$\text{sigmoid}(20g)$	$2m$
$g_1 \cdot g_2$	$2m$

Table 1. Primitive functions $a(g)$ used in the EQL network and the number of times that each primitive function is duplicated in each layer, n_g. m is a hyper-parameter that controls how many times each activation function is duplicated, and thus controls the width of each hidden layer. The sin and sigmoid functions have multipliers inside so that the functions more accurately represent their respective shapes inside the input domain of $x \in [-1, 1]$

2.2. Sparsity Regularization

A key ingredient of making the results of symbolic regression interpretable is enforcing regularization such that the system finds the simplest possible equation that fits the data. In the EQL network, we use sparsity regularization to set as many weight parameters to 0 as possible such that those parameters are inactive and can be removed from the final expression. Thus, the EQL network uses as few nodes as possible, translating into simpler equations that are more readily interpretable and can effectively extrapolate.

Here we compare a smoothed version of $L_{0.5}$ regularization and a relaxed version of L_0 regularization.

$L_{0.5}$ has been proposed in neural networks to enforce sparsity more strongly without penalizing the magnitude of the weights as much as L_1 (Xu et al., 2010; 2012). However, $L_{0.5}$ regularization has a singularity in the gradient as the weights go to 0, which can make training difficult for gradient descent-based methods. To avoid this, we use a smoothed version of $L_{0.5}$ proposed in (Wu et al., 2014), which we label as $L^*_{0.5}$. The $L^*_{0.5}$ regularizer uses a piecewise function to smooth out the function at small magnitudes:

$$L^*_{0.5}(w) = \begin{cases} |w|^{1/2} & |w| \geq a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{1/2} & |w| < a \end{cases} \tag{2}$$

where $a \in \mathbb{R}^+$ is the transition point between the standard $L_{0.5}$ function and the smoothed function. This regulariza-

tion is applied element-wise to all the weight matrices in the EQL network and summed up.

We also use a relaxed version of L_0 regularization introduced in (Louizos et al., 2017). While L_0 regularization is not differentiable and thus not amenable to gradient descent training methods, the relaxed L_0 regularization introduces stochastic gate variables z drawn from hard concrete distributions parameterized by α, which are tuned during training.

The weights $\mathbf{W}$ of the neural network are reparameterized as

$$\mathbf{W} = \tilde{\mathbf{W}} \odot \mathbf{z}$$

where each element of $\mathbf{z}$, $z_{j,k}$, is a stochastic variable drawn from the hard concrete distribution:

$$\begin{aligned} u &\sim \mathcal{U}(0,1) \\ s &= \text{sigmoid}\left(\left[\log u - \log(1-u) + \log \alpha_{j,k}\right]/\beta\right) \\ \bar{s} &= s(\zeta - \gamma) + \gamma) \\ z_{j,k} &= \min(1, \max(0, \bar{s})) \end{aligned}$$

where $\alpha_{j,k}$ is a trainable variable that describes the location of the hard concrete distribution, and β, ζ, γ are hyperparameters that describe the distribution. The regularization penalty is the expectation of z over the distribution parameters, which is given analytically:

$$L_0(\mathbf{W}) = \sum_{j,k} \text{sigmoid}\left(\log \alpha_{j,k} - \beta \log \frac{-\gamma}{\zeta}\right)$$

For the relaxed L_0 regularization, our meta-learning algorithm will also learn the proper settings for α.

2.3. Meta-Learning

Meta-learning aims to design a machine learning model that can adapt well to multiple tasks with little data in each new task that it sees. Here we compare two algorithms: joint training and model-agnostic meta-learning (MAML). In the context of symbolic regression, the tasks correspond to different functions that we want to learn.

2.3.1. Joint Training

Define θ as the collection of parameters in the machine learning model that we aim to learn. In joint training, we sample a function f and perform gradient descent on a batch of data from this function f to reach a new parameter θ'. We then repeat this loop starting from θ' until the training reaches convergence.

During testing, the model is fine-tuned on the new function through gradient descent.

2.3.2. MAML

MAML is a gradient-based meta-learning technique that aims to find a good weight initialization so that the neural network can be adapted to an unseen task with few data points and few training steps (Finn et al., 2017). This is done by training the neural network on multiple tasks during meta-training with a fixed number of steps, and updating the initial weights such that the fixed number of training steps would perform well on the tasks.

Specifically, we start with initial parameters θ and sample a function f. We then perform gradient descent on a batch of data from f to reach parameter ϕ. We then calculate the derivative of the loss at parameter ϕ with respect to the initial parameters θ and perform gradient descent on θ to reach θ'. We throw away ϕ and repeat the process again starting from θ' with a new function f.

MAML differs from joint training in that θ only serves to be a good initialization for learning f, and is not necessarily a good parameter itself to predict f. The assumption is that we perform fine-tuning on θ. This algorithm thus calculates second-order gradients to θ as it is learning how to learn. Raghu et al. (2019) show that MAML can be seen as training a feature extractor common across all tasks and finetuning only the network's last layer to produce task-specific predictions. Thus, in the case of symbolic regression we should be able to meta-learn common operations.

We use the Learn2Learn (Arnold et al., 2020) implementation of MAML. We train using the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 10^{-3}. MAML's inner loop adaptation uses SGD and we experiment with three different learning rates, $\{10^{-3}, 10^{-2}, 10^{-1}\}$, as well as different number of adaptation steps, $\{1, 3, 10\}$. We find that 10^{-2} works best as an inner learning rate and using more than 1 adaptation steps leads to training instabilities.

3. Results: Arbitrary Equations

3.1. Tasks

ID	formula
"id"	$f(x) = x$
"gaussian1"	$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2*\pi}}$
"exp"	$f(x) = \exp(x)$
"sin"	$f(x) = \sin x$
"f1"	$f(x) = x \cdot \sin x - 3$
"f2"	$f(x) = x^2 + 3 \cdot x + 1$

Table 2. Summary of 1D regression tasks to test multi-task symbolic regression with MAML on the EQL network.

We perform symbolic regression on a set of 1-dimensional functions using the EQL network and experiment with varying the number of tasks. We train the EQL network for 10000 outer-loop iterations and consider 100 support and 100 query data points for each regression problem. The regression tasks we consider are summarized in Table 2. We present training loss curves for a small set of target functions and use joint training as a baseline. For all figures in this section the vertical y-axis is the MSE loss and measurements are inter-spaced by 250 update steps over *each* function.

3.2. Regularization

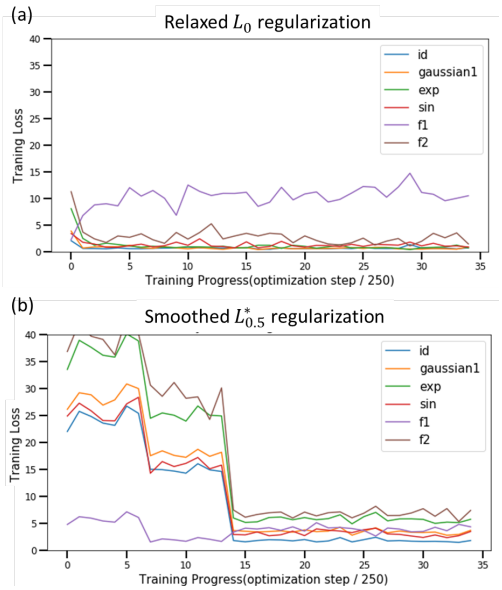

Figure 2. MAML training loss curve on the EQL network with (a) the relaxed L_0 regularization and (b) the smoothed $L_{0.5}^*$ regularization. Note that the EQL network with $L_{0.5}^*$ struggles to find a good initialization in the first few thousand optimization steps, likely as a result of the EQL network getting trapped in a local minima. Sharp improvements are sometimes visible in more than a single task, which suggests that here we notice the intended benefits of jointly training multiple regression tasks. The L_0 regularized EQL network seems to be more stable as it reaches a reasonable initialization much more quickly.

A comparison of the training loss curves for the EQL network trained using MAML are shown in Figure 2. All 6 functions are used during training. The EQL network with the relaxed L_0 regularization quickly finds a good initialization after a few hundred optimization steps and the training loss does not change significantly for the remainder of the training. However, the "f2" function remains remarkably more difficult to fit compared to the rest of the function, suggesting that the EQL network has found a local minima that is suitable for the first 5 functions but not the 6th.

The EQL network with $L_{0.5}^*$ regularization struggles to find a good initialization in the first few thousand optimization steps, although it finds a reasonably good initialization after some time. However, note that the average training loss is still high er than that of the L_0 regularized network.

3.3. Negative Transfer and Training Instabilities

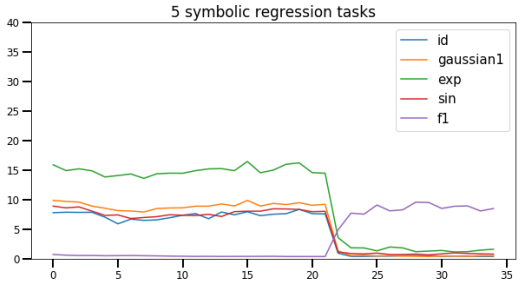

Figure 3. MAML training loss for the $L_{0.5}^*$ regularized EQL network. Applying MAML to EQL can exhibit negative transfer because the performance on "f1" deteriorates while the network learns the other regression tasks.

Figure 3 shows the training curves of the EQL network with the smoothed $L_{0.5}^*$ regularization when the first five functions in Table 2 {"id", "gaussian1", "exp", "sin", "f1"}. The performance on the "f1" function deteriorates at the same time as the network learns to fits the others.

A comparison between Figures 3 and 2(b) (in which the EQL network is trained on all six regression tasks) suggests performance in the multi-task setting is sensitive to the initialization as it can tend to get stuck in local minima do not adapt well to all regularization tasks. For example, we see the EQL network in Figure 3 has found an initialization that performs well in the first four functions, but not the fifth, whereas the EQL network in Figure 2(b) has found an initialization that performs reasonably, but not remarkably, well for all six functions and performs worst for the third and sixth regression functions. Additionally, we find that MAML training heavily depends on hyper-parameters such as both the outer loop's learning rate schedule as well as the inner loop step-size and that higher inner-loop steps-sizes can lead to either exploding loss functions *or* sharp improvements in performance.

3.4. Increasing Network Capacity

We experimented with higher network capacity by doubling the number of units in each hidden layer ($m = 1$ and $m = 2$ and find that the training is more unstable without seeing good performance gains. This supports the intuition that strong regularization is critical to good performance.

3.5. Comparison with Joint Training

Figures 4 and 5 compares the performance of the EQL network trained with either joint training or MAML, for both types of regularization and using various number of training tasks. As above, we find that MAML can be unstable on the $L_{0.5}^*$ regularized EQL network. On the other hand, this same regime leads to the positive jumps in performance that allows MAML to either match joint-training in the three-task scenario or better-fit the train functions in the 5-task scenario.

In the case of the L_0 regularized EQL network, the training is much more stable under MAML. Comparing Figure 4(c) and Figure 5(a) which are both trained on 5 tasks, we see that the type of regularization does not make a huge impact on performance for joint training the EQL network. Comparing joint training and MAML, MAML on average performs better than joint training as it is able to fit most of the training functions extremely well, but it seems to fall into local minima that diminishes the performances of the "f2" task moreso than joint training

4. Results: Function Distributions

4.1. Tasks

The original MAML paper tests the algorithm on a regression task $f(x) = A\sin(\omega x + B)$ where ω is fixed and A and B are sampled for each task. We aim to generalize this and choose tasks from a distribution of formulas. We speculate that equations drawn from the same sub-fields of science and engineering have more in common with each other, analogous to drawing image recognition tasks from similar or different distributions of images. For example, the damped harmonic oscillator describes a wide variety of systems in physics, and is described by the differential equation

$$\frac{d^2x}{dt^2} + \zeta\omega_0\frac{dx}{dt} + \omega_0^2 x = 0$$

The general solutions to this equation are given by $f(t) = a\exp(-vt)\cos(\omega t - \phi)$, where the parameters depend on the parameters in the differential equation as well as the initial conditions. This can also be decomposed into a purely sinusoidal solution, an exponential solution, or a combination depending on the parameters and initial conditions.

Harmonic Oscillator Solution Basis To mimic the distribution of solutions to the damped harmonic oscillator equation, we generate symbolic regression tasks according to a hierarchical generation process:

1. We first sample C from a Bernoulli distribution with $p = 0.5$.
2. We then sample a linear combination of either sinusoidal waves or exponential functions. If C is 0 we sample uniformly $f \in [0.5, 2]$ and $\phi \in [0, 2\pi]$ in order to construct:

$$f^{(1)} = \sin\{2f\pi x + \phi\}$$

Otherwise we uniformly sample $a, b \in [-1, 1]$ to construct:

$$f^{(2)} = a \cdot e^x + b \cdot e^{-x}$$

4.2. Results

In all following results we drastically reduce the number of support and query samples for each regression tasks from 100 data points to just 10 because we can now generate more tasks. This setting better reflects numerous scientific scenarios where gathering observations is either expensive or time-consuming, but we have access to different experiments which stem from the same underlying phenomena.

We train EQL networks using both MAML and joint training on 20 symbolic regression tasks and evaluate on 10 tasks using 10 support and query data points from each function. We use functions generated according to the procedure outlined in Section 4.1 and summarize our results in Figure 6. For this evaluation we did not finetune MAML on the unseen tasks: surprisingly, we find that adaptation hurts final performance using MAML and does not significantly influcence joint training in either direction. The average validation loss is 0.68 for MAML and 0.72 for joint training. In agreement with our findings above we that the validation loss of MAML has a higher variance across different validation tasks. On average MAML outperforms joint training in this low-data regime.

4.3. Out of Distribution Experiments

We evaluate both MAML and joint training on out-of-distribution tasks in order to determine whether the inner loop optimization procedure improves transfer learning lowdate performance. To this end we modify the data generation procedure in Section 4.1 to sample $f \in [2, 4]$ and $a, b \in [1, 2]$ for the validation tasks. Through this experimental section we finetune on new tasks' support sets.

Figure 7 shows how MAML and Joint Training perform on out of distribution tasks as training progresses. We notice joint training converges much faster and outperforms MAML's best performance on some tasks. On the other hand it seems to have reduced generalizability capacities as it is not able to make any progress on the more difficult tasks. MAML on the other hand seems to compromise and perform reasonably well on any task from the unseen domain.

Figure 8 summarizes the final results of EQL methods trained with either method on out of distribution tasks.

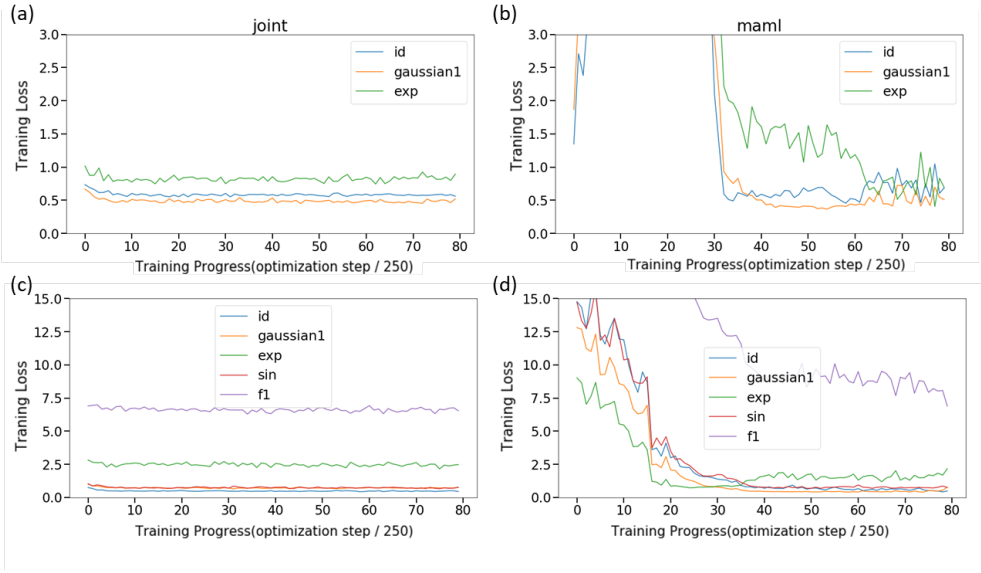

Figure 4. Comparison of (a, c) joint training (b, d) and MAML using either (a, b) three or (c, d) five train tasks. The EQL network uses $L_{0.5}^*$ regularization.

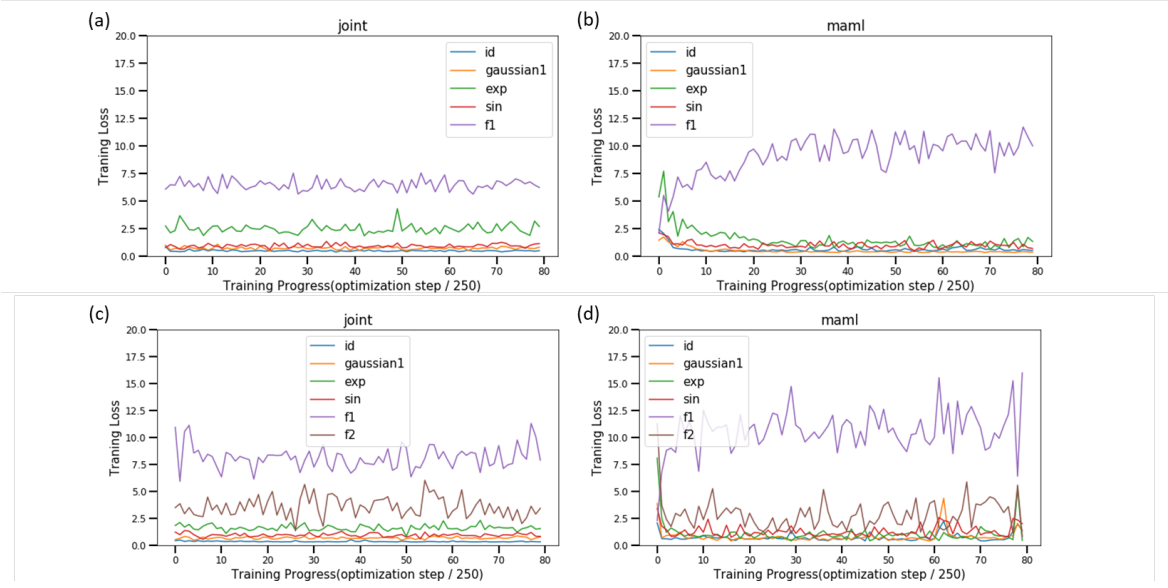

Figure 5. Comparison of (a, c) joint training (b, d) and MAML using either (a, b) three or (c, d) five train tasks. The EQL network uses L_0 regularization.

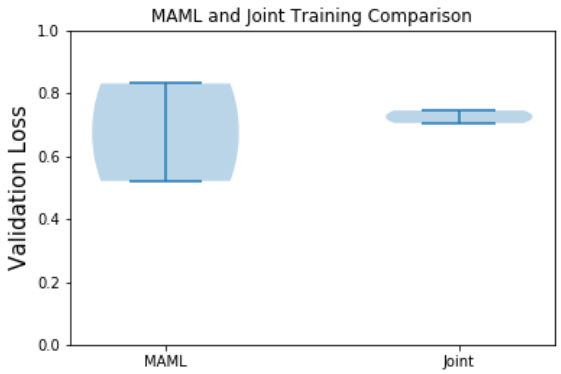

Figure 6. Comparison between EQL networks trained using either MAML or the Joint Training baseline. Both the train and validation tasks are drawn from the same distribution described in Section 4.1. The y axis represents Final validation MSE losses. MAML's performance covers a wider range of values.

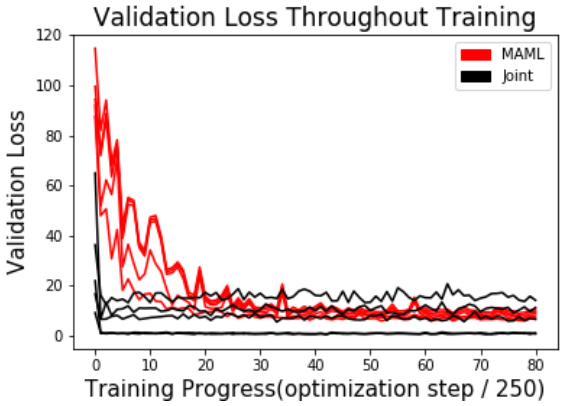

Figure 7. Validation loss throughout training evaluated on out-of-distribution tasks.

MAML's adaptation process makes its final performance distribution more spread out and more robust to distributional shifts in its inputs. MAML's average loss is 8.76 and that obtained from joint training is 6.75. As the differences between train and test tasks distributions' increase using MAML becomes more advantageous.

5. Limitations

5.1. Division

While our prior work on using the EQL network for symbolic regression focused mostly on equations that do not include the division operator (Kim et al., 2020), many equations in physics rely on division. The NALU and EQL$^{\div}$ networks offer two different approaches to integrating the division operator into neural networks (Trask et al., 2018; Sahoo et al., 2018). However, it has been reported that the NALU architecture can be difficult to train and converge (Madsen & Johansen, 2020), so here we focus on replicating the EQL$^{\div}$ approach, which we briefly outline here.

The difficult of using the division operator $f(x_1, x_2) = x_1/x_2$ as-is as an activation function is that it has a pole at $x_2 = 0$ where gradients can explode, and that it is non-monotonic. The EQL$^{\div}$ proposes to use an altered version of the division operator:

$$f(x_1, x_2) = \begin{cases} 0 & \text{if } x_2 \leq \theta \\ x_1/x_2 & \text{otherwise} \end{cases}$$

where θ is a threshold hyperparameter that prevents the function (and the resulting gradient) from becoming too large. Additionally, we introduce a penalty term on the inputs into this function: $p(x_2) = \max(\theta - x_2, 0)$. These two components encourage the network to use only a single branch of the division function to avoid the discontinuity.

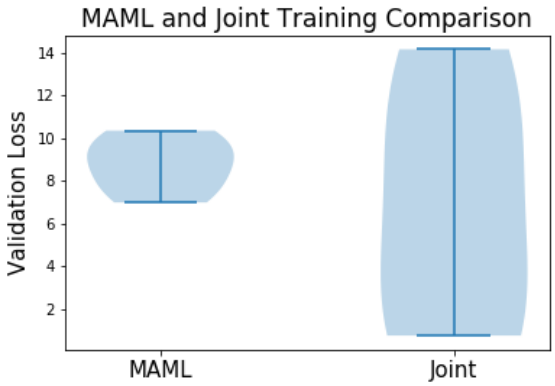

Figure 8. Comparison of MAML and Joint Training performance on out of distribution tasks.

We tried to fit functions $f(x) = 1/x$ and $f(x_1, x_2) = x_1/x_2$, but the EQL network was unable to find a simple equation to fit the data. We also fit the function

$$f(x_1, x_2) = \frac{\sin(\pi x_1)}{(x_2^2 + 1)}$$

which is used for testing in (Sahoo et al., 2018). While the network was able to find this equation, it has a very low success rate and is very sensitive to hyper-parameters. Additionally, the network is prone to exploding gradients, especially when the hyper-parameter θ is set to a small value. Thus, for the results presented here, we do not use the division operator as a basis function, and only fit functions that do not need this primitive. Due to the ubiquity of the

division operator in science and engineering, future work will focus on integrating this into the EQL network.

6. Conclusion

We extend EQL networks to multi-task and low-data regimes as a first step in automating scientific discovery in areas where data is sparse. We empirically test both joint and bilinear training techniques and identify their strengths and weaknesses: the former is often more stable, while the latter often outperforms joint training and is better suited for out-of-domain scenarios. This shows that the EQL network can be adapted towards specific fields or sub-fields of science and engineering without having any explicit inductive biases built in other than the primitive functions. We experiment with two different regularization techniques and identify a convex relaxation of the L_0 norm that leads to simpler and more interpretable symbolic expressions.

References

Arnold, S. M. R., Mahajan, P., Datta, D., Bunner, I., and Zarkias, K. S. learn2learn: A library for meta-learning research, 2020.

Dubčáková, R. Eureqa: software review. *Genetic programming and evolvable machines*, 12(2):173–178, 2011.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv e-prints*, art. arXiv:1703.03400, March 2017.

Kim, S., Lu, P. Y., Mukherjee, S., Gilbert, M., Jing, L., Čeperić, V., and Soljačić, M. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, December 2014.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Madsen, A. and Johansen, A. R. Neural arithmetic units. *arXiv preprint arXiv:2001.05016*, 2020.

Martius, G. and Lampert, C. H. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. *arXiv e-prints*, art. arXiv:1909.09157, September 2019.

Sahoo, S. S., Lampert, C. H., and Martius, G. Learning equations for extrapolation and control. *arXiv preprint arXiv:1806.07259*, 2018.

Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., and Blunsom, P. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pp. 8035–8044, 2018.

Udrescu, S.-M. and Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Udrescu, S.-M., Tan, A., Feng, J., Neto, O., Wu, T., and Tegmark, M. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *arXiv preprint arXiv:2006.10782*, 2020.

Wu, W., Fan, Q., Zurada, J. M., Wang, J., Yang, D., and Liu, Y. Batch gradient method with smoothing L1/2 regularization for training of feedforward neural networks. *Neural Networks*, 50:72–78, feb 2014. ISSN 0893-6080. doi: 10.1016/J.NEUNET.2013.11.006. URL https://www.sciencedirect.com/science/article/pii/S0893608013002700.

Xu, Z., Zhang, H., Wang, Y., Chang, X., and Liang, Y. L 1/2 regularization. *Science China Information Sciences*, 53(6):1159–1169, jun 2010. ISSN 1674-733X. doi: 10.1007/s11432-010-0090-0. URL http://link.springer.com/10.1007/s11432-010-0090-0.

Xu, Z.-B., Guo, H.-L., Wang, Y., and Zhang, H. Representative of L1/2 Regularization among Lq ($0 < q \leq 1$) Regularizations: an Experimental Study Based on Phase Diagram. *Acta Automatica Sinica*, 38(7):1225–1228, jul 2012. ISSN 1874-1029. doi: 10.1016/S1874-1029(11)60293-0. URL https://www.sciencedirect.com/science/article/pii/S1874102911602930.