THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Improving Meta-Learning Performance Through Task Sampling Techniques

JACOB SMITH
SPRING 2021

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Abstract

Meta-learning is a subset of machine learning that involves training a model on a variety of different tasks and testing that model on new tasks with little data per task. Many current meta-learning algorithms employ a naive random sampling approach when selecting the topics to train upon. Domain specific sophisticated methods of sampling tasks exist for domains such as few-shot classification and meta-reinforcement learning which improve the performance of meta-learning models in these domains. Upon review of these methods, it was discovered that the domain of few-shot regression could have a similar task sampling method adapted from these existing methods. After discussing the existing methods for task sampling, a new technique specific to the domain of few-shot regression is introduced. This method is shown to provide a small increase to the performance of meta-learning models on a regression task.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Professor Mehrdad Mahdavi for helping me discover this research topic and exposing me to new ideas in machine learning. I would also like to thank Professor Danfeng Zhang for supporting me throughout my time at Penn State as my Honors Adviser.

Chapter 1

Introduction

This chapter will introduce the idea of meta-learning and how it relates to traditional deep learning methods. The task selection methods of many current meta-learning models will be discussed. Finally, sophisticated methods of task selection for the domains of classification and reinforcement learning will be introduced.

1.1 Meta-Learning

Deep learning has become a ubiquitous topic in recent years. Methods employing deep learning have been developed to solve a wide variety of problems including language processing [3], image recognition [4], and recommender systems [5]. However, these methods all share a significant flaw. Deep learning methods require a very large amount of data to produce meaningful results [6]. An often stated goal of artificial intelligence research is to model as closely as possible the way in which humans learn information. In contrast to traditional deep learning methods, humans can learn new information with a relatively small number of examples. The desire to model this behavior is what gave rise to the idea of few-shot learning, generating a model to perform a task using only a few examples. One of the best performing techniques in solving few-shot learning problems today is the idea of meta-learning, or learning to learn.

In a meta-learning problem, a model is trained on a variety of different tasks $\mathcal{T}_i$ during the meta-training phase. Instead of trying to learn specific solutions to these tasks, a meta-learning model is tasked with learning information about the structure and common features of these tasks in order to learn these tasks with fewer examples. In other words, the model learns the most effective learning techniques across the meta-training tasks. At meta-test time, the model is presented with a new unseen task and only a few examples relating to that task. Using these few examples combined with the prior knowledge of learning techniques, the model must then learn to perform this task.

One example of such a meta-learning algorithm is Model-Agnostic Meta-Learning, or MAML [7]. In this algorithm, a batch of tasks is first sampled from a uniform distribution over all possible tasks. Then, starting from global parameters θ, one step of gradient descent is performed on each of these tasks resulting in new parameters θ_i' for each task $\mathcal{T}_i$. Finally, one step of gradient descent is performed using the sum of each task's loss function when supplied with these new parameters θ_i'. This step of gradient descent is used to update the global parameters θ. The goal of this algorithm is to discover an initial parameterization θ that allows for quick convergence after a small number of gradient descent steps over a wide variety of tasks.

1.2 Task Selection

At their core, meta-learning methods rely heavily on the sample tasks chosen for meta-training. Much like deep learning models rely heavily on a large amount of data, meta-learning models require a large amount of different tasks upon which to train. While the meta-test tasks are unknown, it is important that the training tasks be representative of the tasks that will be seen at meta-test time.

Vanilla meta-learning methods often employ a strategy of uniform random sampling to generate the tasks used for meta-training [7] [8]. These sampling methods can still lead to impressive results in few-shot learning problems, but a random sampling technique ignores the structure of tasks and the relationship between tasks. For example, a random sampling technique across the distribution of possible meta-training tasks might generate a set of tasks that are all very similar. When training on these tasks, the meta-learning model will generate a strong learning technique that will allow it to adapt very quickly to this set of tasks. However, if the tasks encountered at meta-test time are significantly different from the training tasks, the model may not be able to generalize well. In addition, a random sampling technique may generate meta-training sets that contain a number of very easy tasks. Although the model would perform very well on these tasks, they provide very little information and thus will not significantly improve the model's ability to generalize to more difficult tasks. In ignoring the structure and relationships between tasks, naive random sampling techniques hamper the ability of meta-learning models to generalize well to unseen test cases.

1.3 Current Methods of Task Selection

Despite the reliance of many meta-learning models on random sampling techniques, there exist some more sophisticated methods of task sampling for meta-learning problems. One example of such a method is the Adaptive Sampling method [2]. This method of task sampling works in the classification domain. Tasks in this domain correspond to a set of classes, and the Adaptive Sampling method takes into account the relative difficulty of discriminating between the classes in a given task to select a task for meta-training. Another example is Information-Theoretical Task Selection [1]. This method of task selection works in the reinforcement learning domain and takes into account the similarity between tasks in the training set and the relevance of these tasks to a validation set to determine which tasks to use for meta-training.

Both of these task selection methods can improve performance in their respective domains, but these methods both rely on the specific structure of these domains. Therefore, these selection techniques cannot be easily adapted to other domains like the domain of few-shot regression. The problem of few-shot regression does not yet have a specific sampling technique for improving the performance of meta-learning models. After a discussion of the two aforementioned techniques and the principles guiding them, a sampling technique for this domain based upon the lessons learned from existing techniques will be introduced.

Chapter 2

Previous Work

This chapter will detail two existing task sampling methods for meta-learning, Adaptive Sampling and Information-Theoretic Task Selection. These methods are designed for the classification and reinforcement learning domains, respectively. The intuition behind both techniques as well as their specific implementations and the results they achieve over vanilla meta-learning models will be discussed.

2.1 Adaptive Sampling

This section will introduce the Adaptive Task Sampling method [2] for meta-learning models. The problem setting for few-shot classifcation problems, the setting in which Adaptive Sampling operates, will be introduced. The methodology behind the Adaptive Sampling technique will be discussed. Finally, the results of implementing this technique on classification problems will be shown and discussed.

2.1.1 Problem Setting

The Adaptive Task Sampling method is designed to improve the performance of meta-learning models on few-shot classification problems [2]. These problems are characterized by two datasets, a meta-training dataset $\mathcal{D}_{train}$ and a meta-test set $\mathcal{D}_{test}$. Each of these datasets has a respective set of classes $\mathcal{C}_{train} = \{1, 2, \dots, n\}$ and $\mathcal{C}_{test} = \{n+1, n+2, \dots, n+m\}$. Typically, the meta-training dataset has a large number of both classes and examples. The class sets for the two datasets are disjoint. Each task $\mathcal{T}_i$ used by the meta-learner in this problem is a K-way-N-shot classification task formed by sampling K different classes $\mathcal{L}$ from $\mathcal{C}_{train}$ and N examples from each of these classes. The goal of a meta-learning agent on this problem is to use these tasks to learn a model on $\mathcal{D}_{train}$ that can generalize using a small number of training steps to $\mathcal{D}_{test}$.

Many existing meta-learning models rely on an episodic training strategy to solve this problem [9]. In this strategy, each sampling of classes $\mathcal{L}$ corresponds to one episode. Once the classes have been sampled, a support set $\mathcal{S}$ and a training batch $\mathcal{B}$ are sampled from $\mathcal{D}_{train}$, both consisting of labeled examples from the classes in $\mathcal{L}$. The model then predicts the labels of the examples in $\mathcal{B}$ given the examples in $\mathcal{S}$. The objective function the training is attempting to minimize is as follows:

$$l(\theta) = \mathbb{E}[-\sum_{(x_n, y_n) \in \mathcal{B}} \log p_\theta(y_n | x_n, \mathcal{S})],$$

where θ are the parameters for the model and $p_\theta(y_n|x_n, \mathcal{S})$ is the probability of the model predicting the correct label y_n given the example x_n and the support set $\mathcal{S}$.

2.1.2 Sampling Method

The intuition behind the Adaptive Task sampling method is that certain class-pairs are more difficult to differentiate between than others. This method forces a model to learn from more difficult tasks, which provide more information to the model than simple tasks [2]. This is accomplished by keeping track of which class-pairs are the most difficult to discriminate between during the training phase. The Adaptive Sampling method characterizes this class-pair difficulty for a pair of classes (i, j) as the average probability that an example belonging to one of these classes is mistakenly classified as being in the other. Formally, this probability is

$$\bar{p}((i,j)|\mathcal{S}, \mathcal{B}) = \frac{\sum_{(x_n, y_n = j) \in \mathcal{B}} p(c = i | x_n, \mathcal{S})}{N} + \frac{\sum_{(x_n, y_n = i) \in \mathcal{B}} p(c = j | x_n, \mathcal{S})}{N}.$$

The adaptive sampling algorithm uses these probabilities to update a set of class-pair potential functions $C(i, j)$ during meta-training. These functions increase in proportion to a greater $\bar{p}((i,j)|\mathcal{S}, \mathcal{B})$, and thus capture the notion of difficulty in discriminating between specific class-pairs. The class sampling $\mathcal{L}$ is selected based upon those class-pairs with the greatest $C(i, j)$, crafting a set of classes that represent the most difficult pairwise classification problems in the meta-training data. In this way, the model avoids training on simple tasks which would not provide as much information.

2.1.3 Results and Analysis

Table 2.1 shows the results of the class-pair based Adaptive Sampling method against other meta-learning algorithms on the miniImageNet [9] dataset. This dataset consists of 100 classes with 600 84×84 images per class. The dataset is divided into 64 meta-training classes, 16 meta-validation classes, and 20 meta-test classes. Table 2.1 shows the mean accuracy and 95% confidence intervals on the meta-test set.

Methods	5-way-1-shot	5-way-5-shot
Matching Network [9]	43.44 ± 0.77	55.31 ± 0.73
MAML [7]	48.70 ± 1.84	63.11 ± 0.92
PN [10]	59.25 ± 0.64	75.60 ± 0.48
PN with gcp-sampling [2]	61.09 ± 0.66	76.80 ± 0.49
MetaOptNet-RR [10]	61.41 ± 0.61	77.88 ± 0.46
MetaOptNet-RR with gcp-sampling [2]	63.02 ± 0.63	78.91 ± 0.46
MetaOptNet-SVM [10]	62.64 ± 0.61	78.63 ± 0.46
MetaOptNet-SVM with gcp-sampling [2]	64.01 ± 0.61	79.78 ± 0.47

Table 2.1: Results of the adaptive sampling method (gcp-sampling) compared to other few-shot classification tasks, adapted from [2].

These results show that the Adaptive Sampling method yields a significant improvement in meta-test accuracy over vanilla meta-learning models. These results support the idea that focusing specifically on difficult tasks, or tasks on which the model has a higher probability to fail, results in more meaningful learning.

The main drawbacks in considering this method of task sampling for regression are the reliance on the classification setting and the number of possible class-pairs. For this method, a matrix of size $|\mathcal{C}_{train}| \times |\mathcal{C}_{train}|$ must be maintained and updated after each task selection episode to capture all of the class-pair potential functions. For a meta-learning classification problem with a reasonable number of classes like miniImageNet, which contains 64 different training classes, this does not pose an issue. However, as the number of possible classes increases, the amount of work required to maintain this matrix increases by $O(n^2)$. With this in mind, consider a direct transfer of this method to a regression problem. Since this setting has no distinct classes, the relative difficulty of something else must be measured. Regression problems involve assigning predicted outcomes to individual input points, so maintaining a set of potential functions for different input points or sets of input points would be analogous to the classification case. In the regression setting, input points are sampled continuously from a certain range. Clearly, maintaining potential functions for each possible input on this continuous range would be intractable. Therefore, a different heuristic for task selection must be considered.

2.2 Information-Theoretic Task Selection

This section will explore the Information-Theoretic Task Selection method [1] for meta-reinforcement learning models. The problem setting and notation for reinforcement learning problems will be introduced. The specifics of the task sampling method will be explained. Finally, the results of using this sampling method in meta-reinforcement learning tasks will be shown and discussed.

2.2.1 Problem Setting

The Information-Theoretic Task Selection (ITTS) method [1] is designed to improve performance on the domain of Reinforcement Learning. A task in reinforcement learning is represented as a Markov Decision Process $m = \langle S, A, R, P, \gamma, \mu \rangle$. In a Markov Decision Process, S represents the set of possible states, A represents the set of actions, $R \subset \mathbb{R}$ is the set of rewards, $P(s', r|s, a)$ represents the joint probability distribution of the next state and the reward given the current state and the action taken, $0 \leq \gamma \leq 1$ represents a discount factor, and $\mu(s)$ is the distribution of initial states. An agent in a Reinforcement Learning scenario is represented by a policy $\pi(a|s)$, which represents the probability of taking a certain action given the current state of the agent. The goal of this agent is to learn an optimal policy π^* which maximizes the expected total reward $G_t = \mathbb{E}[\sum_{i \geq t} \gamma^{i-t} r_{i+1}]$ starting from a state s_t at time t, where r_{i+1} is the reward resulting from the action taken at time i.

In the context of meta-learning, each task $\mathcal{T}_i$ corresponds to a single Markov Decision Process sampled from a distribution $p(\mathcal{M}_{train})$ where $\mathcal{M}_{train}$ is the meta-training set. Throughout meta-training, the meta-learning agent learns a policy π that is transferred to a new task sampled from $\mathcal{M}_{test}$ at meta-test time. Since the model must transfer its policy to each new task, it is required that a policy for one Markov Decision Process m be defined for every possible state and action of the other processes in $\mathcal{M}_{train}$ and $\mathcal{M}_{test}$.

2.2.2 Sampling Method

The intuition behind ITTS is that, for a task to be useful to the meta-learning model, it must be both different enough from other training tasks and yet relevant enough to transfer well to other tasks. To accomplish this, ITTS samples from the meta-training set $\mathcal{M}_{train}$ two disjoint sets, a training set $\mathcal{T}$ and a validation set consisting of K tasks $\mathcal{F}$. True to its name, ITTS defines the concept of difference and relevance from an information-theoretic perspective [1]. This method defines the concept of difference between two tasks m_1 and m_2 as the average Kullback-Liebler divergence between the optimal polices for these tasks over the states of the validation tasks. This divergence is a measure of difference between two probability distributions. It acts as a measure of the amount of information in one set that can be used to differentiate it from the other [11]. The equation as used by ITTS is as follows:

$$\delta m_1, m_2 := \frac{1}{K} \sum_{m_j \in \mathcal{F}} \frac{1}{|S_j|} \sum_{s \in S_j} \sum_{a \in A} \pi^*_{m_1}(a|s) \log \frac{\pi^*_{m_1}(a|s)}{\pi^*_{m_2}(a|s)}.$$

ITTS defines the relevance of a task m_1 to another task m_2 in terms of the optimal policy $\pi^*_{m_1}$. In order to calculate the relevance, first l reinforcement learning steps are performed on m_2 starting from $\pi^*_{m_1}$. Relevance is then calculated using the expected difference in entropy between the optimal policy for m_1 and the newly calculated policy $\pi^l_{m_1,m_2}$ over the states of the validation tasks. Formally, the relevance measure is as follows:

$$\rho_l(m_1, m_2) := \mathbb{E}[H(\pi^l_{m_1,m_2}(a|s)) - H(\pi^*_{m_1}(a|s))].$$

As opposed to Adaptive Sampling, which runs as part of the meta-training phase, ITTS creates a sample of training tasks before meta-training begins. As stated above, ITTS begins by taking two separate sets of tasks, $\mathcal{T}$ and $\mathcal{F}$, and an empty set of tasks $\mathcal{C}$, which is the set of selected meta-training tasks. Then, each task in $t \in \mathcal{T}$ is compared to each task in $\mathcal{C}$ using the above difference measure. If this new task is sufficiently different each task in $\mathcal{C}$, then it is compared to each task in $\mathcal{F}$ using the relevance measure. If the task t is deemed to be relevant to at least one task in $\mathcal{F}$, i.e. the expected entropy between π^*_t and $\pi^l_{t,f}$ is below some threshold for some $f \in \mathcal{F}$, then the task t is added to $\mathcal{C}$ and the next task is considered.

2.2.3 Results and Analysis

Figures 2.1 and 2.2 show the performance of ITTS on two separate locomotion domains from [7]. The "Ant" domain consists of a 3D quadruped and the "Cheetah" domain consists of a 2D cheetah. The reward functions for both domains consist of the negative absolute value between the velocity of the agent and a randomly selected goal point [7]. Thus, the rewards increase as the velocity of the agent increase towards the goal point.

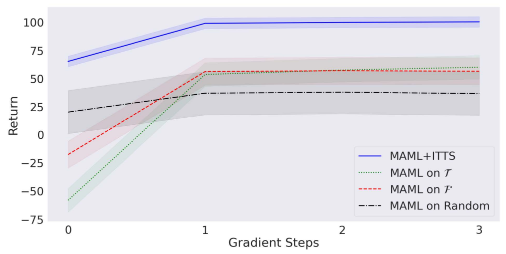

Figure 2.1: Results of the ITTS method on the Ant domain from [1]

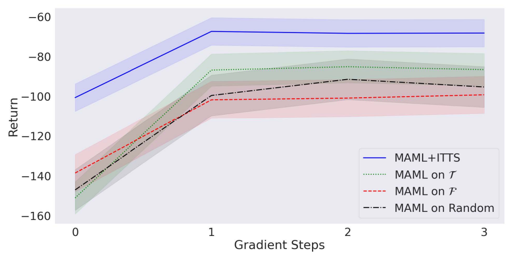

Figure 2.2: Results of the ITTS method on the Cheetah domain from [1]

As with Adaptive Sampling, these results once again show that the adoption of a sophisticated sampling strategy can lead to significant improvements in meta-test accuracy over vanilla meta-learning models. Going further, the intuition from this technique of ensuring that the sampled tasks are different enough from each other and relevant enough to the overall problem seems to be much more readily adaptable to a regression setting.

The drawback is once again that this method cannot be directly transferred to a regression setting. While the intuition behind this technique may generalize more readily, the specific implementation of measures for difference and relevance in this method cannot be transferred to the regression domain. Therefore, although the idea behind this approach seems promising, a new measure for the concept of difference must be employed before this method can be adapted for regression.

Chapter 3

Task Selection for Regression

This chapter introduces the problem of task selection for the meta-learning of regression tasks. The setting of a regression problem is formally laid out. Then, the respective approaches to task sampling of Adaptive Sampling and ITTS are considered to form a basis for a method of task selection for regression problems. Finally, an algorithm is introduced which extends the ideas present in Adaptive Sampling and ITTS to sample tasks for a meta-learning regression setting.

3.1 Problem Setting

In the meta-learning setting for regression, the problem is characterized by two datasets, a meta-training set $\mathcal{D}_{train}$ and a meta-validation set $\mathcal{D}_{test}$. Each of these datasets contain functions f_i. These functions are required to be continuous. A single task $\mathcal{T}_i$ in this domain corresponds to K-shot regression problem on a single function f_i. From this function, K input-output pairs (x_j, y_j) are sampled such that $f_i(x_j) = y_j$. The goal of a model is to take as input the K inputs sampled from $\mathcal{T}_i$ $(x_1, \ldots, x_K)$ and predict the corresponding outputs $(y_1, \ldots, y_K)$. A common loss function used for optimization is the mean-squared error:

$$l(\theta) = \sum_{x_j, y_j \in \mathcal{T}_i} ||f_\theta(x_j) - y_j||_2^2,$$

where θ are the parameters of the model and f_θ is the function used by the model to predict the output. The goal of a meta-learning model is to learn a model on $\mathcal{D}_{train}$ that can generalize well to functions in $\mathcal{D}_{test}$ given only a small number of input-output pairs K.

3.2 Sampling Method

After reviewing both the methods of Adaptive Sampling and ITTS in their respective domains, a framework on which to build a sampling technique for regression becomes apparent. Both of these techniques focus on the idea that repeatedly training on similar or easy tasks will lead to slower learning. Thus, a task sampling method for regression must focus on sampling functions from the test data that provide more information to the model.

It is difficult to enforce a measure of difficulty on the sampling of tasks for regression. This stems from the fact that input-output pairs are only sampled after a function has already been selected. The difficulty of a few-shot regression task can depend heavily on the way these points are sampled, as a sample of points that span the domain upon which the function is being predicted can lead to a much easier problem than a small cluster of points in one area on the same function. Coupled with the fact that a direct transfer of the difficulty measurement for Adaptive Sampling to the regression setting would be intractable, it is more useful to focus on a notion of difference between functions.

Enforcing difference between the functions being sampled ensures that the meta-learning model does not simply learn to perform well on a particular subset of the functions in $\mathcal{D}_{train}$, but upon the entire range of functions this set represents.

To use the difference between functions as a parameter for sampling tasks, a concrete measure of the difference between functions must be employed. Since the functions in a regression problem are required to be continuous, they can be understood through the lens of the L^2 space. The L^2 space is a vector space on functions which provides the L^2 norm:

$$d(f,g) = (\int_a^b (f(x) - g(x))^2 dx)^{1/2}, \tag{3.1}$$

where f and g are two separate functions and $[a, b]$ is the interval on which points are being sampled.

This norm defines a measure of distance between two functions. Using this norm, functions can be sampled based upon their difference between previously sampled functions to ensure that the model does not repeatedly train on very similar functions which may inhibit the model's ability to generalize to new functions.

3.3 Algorithm

Using the measure of distance between two functions presented in the previous section and the ITTS implementation of enforcing difference between selected tasks as a guideline, the following algorithm for sampling tasks for meta-learning on regression is presented:

Algorithm 1 Task Selection for Regression

```
Require: Meta-training data $\mathcal{D}_{train}$, tolerance $\varepsilon$
 1: $T \leftarrow \{\}$
 2: for Training function $f$ in $\mathcal{D}_{train}$ do
 3:     unique $\leftarrow true$
 4:     for Function $t$ in $T$ do
 5:         Calculate $d(f, t)$ according to (4.2)
 6:         if $d(f, t) < \varepsilon$ then
 7:             unique $\leftarrow false$
 8:         end if
 9:     end for
10:     if unique then
11:         Add $f$ to $T$
12:     end if
13: end for
14: return $T$
```

Algorithm 1 is to be run before the meta-training phase. It takes as input the entire set of meta-training data and a tolerance ε which corresponds to the minimum distance between functions to be trained upon. This tolerance can either remain fixed throughout the task selection process or begin large and decrease over time as more functions are checked. It outputs a set T of training tasks that are sufficiently different according to the L^2 norm presented in the previous section. The set T is then used as the meta-training set for the meta-learning algorithm. It builds the set T by comparing each training example from $\mathcal{D}_{test}$ to the currently selected tasks in T. If a task is sufficiently different from each task currently in T, it is added to T.

Chapter 4

Experimental Results

In this chapter, an experiment to show the efficacy of the Regression Task Sampling method introduced in the previous chapter is expressed in detail. Then, the results of this experiment are shown. The results suggest that Regression Task Sampling can lead to an improvement of generalization performance over vanilla few-shot regression solutions.

4.1 Experiment Setup

For these experiments, Algorithm 1 was applied to MAML [7]. As a result, the same parameters used in the MAML experiments will be adopted here. Each function in $\mathcal{D}_{train}$ is a sine wave with random amplitude and phase. The range of possible amplitudes is $[0.1, 5.0]$, and the range of possible phases is $[0, \pi]$. The K inputs $x_1, \dots, x_K$ are sampled uniformly from the interval $[-5.0, 5.0]$. The loss used is the mean-squared error as specified in section 4.1. The model used is a neural network with 2 hidden layers of size 40 using the ReLU activation function. The model trained using MAML takes $K = 10$ examples per meta-update and uses a fixed step size of $\alpha = 0.01$. The model is optimized using Adam [12].

In this experiment, the model using the regression task selection method is compared against two different baselines. The first is MAML with uniform task selection, which uses the same parameters as above. The second is a pretrained model, which simply trains to regress to a random sinusoid and is then fine tuned on the $K = 10$ points at test-time. All tests were run using the code implemented in MAML [7]. Code for this experiment is available at: `https://github.com/jqs6452/Regression-Task-Selection`.

4.2 Results

Table 4.1 shows the results of this experiment. For each model, the average loss after each gradient step at meta-test time is recorded.

	0	1	2	3	4	5	6	7	8	9	10
Pretrained	3.07	2.92	2.82	2.74	2.68	2.62	2.56	2.52	2.48	2.44	2.40
MAML	3.10	0.41	0.22	0.17	0.15	0.13	0.12	0.12	0.11	0.11	0.11
MAML with Regression Task Sampling	3.12	0.34	0.18	0.14	0.12	0.11	0.10	0.10	0.09	0.09	0.09

Table 4.1: Results of the experiment comparing average loss after each gradient step at meta-test time for a pretrained baseline, MAML with uniform sampling, and MAML with Regression Task Sampling

These results show that MAML with Regression Task Sampling leads to slightly lower losses after the first update step and on all subsequent update steps. This suggests that the Regression Task Sampling method has led to better generalization performance by the model, albeit only slightly. Once possible explanation for the small magnitude of the performance increase is that the specific task used in this experiment is too easy to see a large improvement. After only one gradient step, MAML already reaches a very low loss at meta-test time. As a result, any improvement offered by task selection will seem small in comparison to the increase in performance MAML alone provides over the classical pretrained model. Regardless, the results suggest that Regression Task Sampling offers some improvement over vanilla MAML, and this improvement may become more obvious with more difficult regression tasks.

Chapter 5

Conclusion

5.1 Benefits

The greatest benefit of the new Regression Task Selection method is its apparent improvement in performance over vanilla meta-learning models for regression. As shown in the experimental results, applying this method of task selection to the MAML model led to slight improvements in generalization accuracy for K-shot regression problems.

Another benefit of this approach is in its ease of implementation. As shown in the appendix, Regression Task Selection can be implemented in only a few lines of code. In addition, this method makes no assumptions about the underlying learning model. As a result, this method can be implemented on any model for meta-learning regression tasks.

5.2 Drawbacks

The largest drawback of this method is that the performance gains realized are minimal on simple regression tasks. Simply put, existing meta-learning models already perform very well on simple few-shot regression tasks. Thus, there is little room for improvement on these simple examples, regardless of the efficacy of the task sampling technique. However, the improvement offered by Regression Task Selection is suggested to exist by the experiments, and this improvement in performance may be even greater when applied to a more difficult regression problem.

Another drawback of this technique is the increase in computation complexity imposed by the method. As shown in Algorithm 1, Regression Task Selection requires that an integral be calculated each time a new function is compared with one already in the output set. Although this algorithm has no effect on computation time during meta-testing, it does increase the amount of time spent in meta-training.

5.3 Future Work

The topic of task selection for meta-learning is an exciting one. One of the most definitive aspects of human intelligence is the ability to learn from only a few examples, and efficient task selection techniques can improve the performance of meta-learning models on such few-shot learning problems. Given the relatively small amount of task selection algorithms available today, the topic is wide open for future work to create new sampling methods that can improve generalization performance even further.

Perhaps one of the most interesting topics for future research is finding a task selection method that does not make any assumptions on the underlying structure of tasks. All of the methods discussed in this work require prior knowledge of the task domain and cannot be easily transferred to other task domains. Thus, the discovery of further sampling techniques that can use similar notions of difference, relevance, and difficulty as considered by the methods discussed in this work could lead to even better few-shot learning performance across many domains.

Bibliography

[1] Ricardo Luna Gutierrez and Matteo Leonetti. Information-theoretic Task Selection for Meta-Reinforcement Learning. *arXiv e-prints*, page arXiv:2011.01054, November 2020.

[2] Chenghao Liu, Zhihao Wang, Doyen Sahoo, Yuan Fang, Kun Zhang, and Steven C. H. Hoi. Adaptive task sampling for meta-learning. In *Proceedings of the 16th European Conference on Computer Vision*, August 2020.

[3] Marc Moreno Lopez and Jugal Kalita. Deep Learning applied to NLP. *arXiv e-prints*, page arXiv:1703.03091, March 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[5] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In Christopher Burges, Léon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems 26 (2013)*, volume 26, page 9. Neural Information Processing Systems Foundation (NIPS), 2013.

[6] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153:46–53, 2018.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[8] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[10] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Jacob Smith

Summary

Motivated computer science student with experience in software development and algorithm design. Excited to collaboratively apply advanced techniques like artificial intelligence to modern software and cybersecurity problems.

Education

The Pennsylvania State University – Schreyer Honors College
Major: Computer Science August 2017 – Present (2021)
Relevant Coursework:

- Artificial Intelligence
- Operating Systems
- Data Structures and Algorithms
- Systems Programming

Experience

Freelance Software Developer May 2018 – August 2018

- Developed an exercise data and injury prediction platform for La Salle University
- Used Java Servlets, JSP, and JavaScript
- Administered a SQL database for this platform
- Used cloud development technology for an efficient development cycle
- Trained staff on using this platform

SAP STEM Innovators, Newtown Square, PA July 2017 – August 2017

- Worked collaboratively in an agile development environment
- Oversaw the SAP Professional Orientation program
- Designed an educational Internet of Things project
- Developed a music classification app using open-source APIs

Skills

- Java, Python, C and C++
- Familiarity with SQL and JavaScript
- Familiarity with AI-based natural language processing and generation
- Agile development and design thinking methodology
- Algorithm design and implementation