Clustering Mixed Task Distributions using Meta-Learners

Suraj Menon, Wei Da

1 Abstract

We propose an algorithm in this paper that uses solely meta-learner instances to cluster data from mixed datasets. Previous work on meta-learning(such as MAML [FAL17]) has shown success in few-shot image classification with datasets such as Omniglot and Fungi when training on each of the datasets individually. An assumption in these models is that these datasets are able to perform meta-learning due to the common structure that exists between the many classes or 'tasks' within the datasets, but which would not expect to generalize to classes outside of this 'task-distribution'. We consider the fluidity of this definition of 'task-distribution' and attempt to discern to what degree MAML is truly able to concretely recognize such a concept. To do this, we consider attempting to train, without any task-distribution labels, a dataset mixed from two different task-distributions (in this paper the Omniglot and Fungi dataset), and attempt to see if the natural bias that comes through clever training of a 2-MAML-pipeline model (2 separate but simultaneously trained MAML instances) allows for successful clustering of the data.

The theory behind this is to extend the inference methodology from [GRA+18] to an added random variable γ which acts as a 'meta-distribution' over our MAML pipelines. In our work we mainly focus on a mixed-dataset with two task-distributions. In this way, we can consider having a parameter γ for each class in the mixed distribution dataset, and that each $p(\theta_i|\gamma)$ as being Bernoulli distributed (with probability p of belonging to a MAML instance 1, and a probability (1-p) of belonging to a MAML instance 2).

We propose an algorithm that is able to successfully cluster this dataset to about 70 percent accuracy after careful tuning of new hyperparameters. Though the clustering of the data does not reach the same level of accuracy as a baseline K-Means method, we do see comparable accuracy when retraining the clustered datasets on a single MAML pipeline. We consider from these results the larger question of what any particular algorithm like the one we propose may say about how 'different' any two task-distributions are, and what that means in considering what might be the optimal amount of 'shared parameters' between any two meta-learning models.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Introduction

Classical supervised learning usually target to train a model over single training dataset and hopefully get small error on the test dataset.

Unlike supervised learning problem, meta learning is designed to train across multiple datasets(tasks) ($\mathscr{T}_1, \mathscr{T}_2, ...\mathscr{T}_n$) and target to capture share structures between these datasets(tasks). So, when a new dataset $\mathscr{T}_{n+1}$ coming, the learned structure can help to build model even with few training examples.

So, the key assumption of meta learning problem is training and validation tasks are driven from the same task distribution and have share structure, which means $\mathscr{T}_1, \ldots, \mathscr{T}_n \sim p(\mathscr{T}), \mathscr{T}_j \sim p(\mathscr{T})$ (Finn, Hausman, et al. 2020)

In this paper, we try to explore breaking (or relaxing) this assumption. By experiments, we confirm meta learning models would work poorly on mixed datasets from two different distributions . Then we propose two approaches with unsupervised learning techniques to resolve this issue with theoretical analysis.

3 Related Work

Meta Learning, which is designed to allow machine quickly learn new task with few examples by share knowledge across old task, has been applied into multiple supervised, reinforcement learning and unsupervised learning applications Hsu, Levine, and Finn 2018.

There exists three basic approaches for meta learning: 1) Black-box adaptation with recurrent neural network(Santoro et al. 2016, Garnelo et al. 2018), 2) Optimization-Based approaches (MAML, Finn, Abbeel, and Levine 2017) 3) non-parametric approaches(Snell, Swersky, and Zemel 2017)

There exists couple of works try to incorporate Bayesian approach (parameter distribution) with meta learning to reduce "task ambiguity" (Finn, Xu, and Levine 2018, Ravi and Beatson 2019, Grant et al. 2018) Besides these, instead of capture global share knowledge and structure, Yao et al. 2019 group tasks into sub-clusters and capture the cluster-level's structure for adapting new incoming tasks.

We select MAML as the basic meta learning algorithm and try to split the mixed dataset by offline clustering and online inferencing,which are inspired by Bayesian and clustering approaches.

4 Dataset and Methods

4.1 Data Preparation

Triantafillou et al. 2019 has provided couple of meta datasets and provide framework to transfer original data into TensorFlow Record and running meta algorithm over them.

We select Omniglot(1623 classes in total) and FGVCx Fungi(1394 classes in total) as our target dataset by normalizing all image into 28x28x3 and then randomly mixing classes together.

We further use a train set of .70 of all available classes, and furthermore .10 of all classes for the validation set and .20 for the test set. This applies for all the experiments and test results in the section later.

4.2 K-Means Pipelines

We firstly consider to use pre-trained clustering model to split data points for each meta learning pipelines during online training. We assume the distribution and pattern of two dataset are very unlike and can be easily detected by clustering.

We choose K-means to cluster the whole mixed training data and get two centroids for each clusters. Then for epoch, before training MAML models, the data point would be fed into the pipeline whose centroid is closer to.

Finally, we see pretty good results with this simple approach to training model on the mixed dataset (with slight performance drop compared with training each dataset separately).

4.3 Inference Method

Though K-Means provides a useful way to understand structural information about our mixed dataset before training on MAML, it does not give us information on the limits of how MAML will interpret the mixed 'task-distributions' since it uses no information from MAML itself to do the clustering.

Though our theoretical final goal would be to have a 'MAML of MAMLs' of sorts, meaning an end to end trained meta learner of meta learners, we consider instead the simpler task of attempting to classify each given 'class' or 'task' from our initial mixed dataset while in a totally unsupervised setting (meaning no pre-training of either MAML pipeline and no K-Means separation of the mixed dataset). We consider the idea that by understanding how successful this process can be, we can begin to intuit how distinctive any two 'task-distributions' are when considered under the lens of MAML training.

For initial intuition, we look to the Probabilistic interpretation of MAML as described in [GRA+18]. We use the intuition understood here that the meta-learning problem is aiming to understand a common structure among some distribution $p(\mathscr{T})$, and so we assume than in our ideal model made to work with multiple different distributions in the style of $p(\mathscr{T})$ that we will likely need to train a different meta-learner for each 'task-distribution'. We then assume that MAML can be thought of as an approximation of the probability graphical structure in Figure 1. For our purposes, we consider then another random variable γ, upon which influences what is the usual meta-parameter variable θ. In the purest theoretical sense, the distribution γ would contain higher levels parameters that which influence the multiple instances of θ, each which represent individual 'task-distributions'. For our purposes in which we only consider two datasets mixed together, $p(\theta_m|\gamma)$ can be thought of as a set of Bernoulli random variables which dictate the probability of a a generated example belonging to either of the meta-learning pipelines.

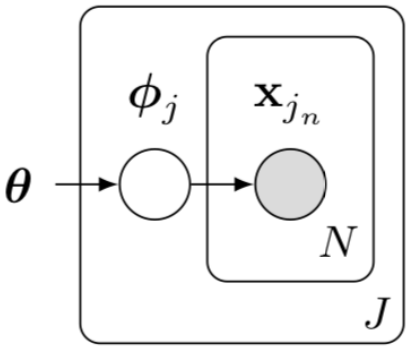

Figure 1: Graphical Model of MAML from [GRA+18]

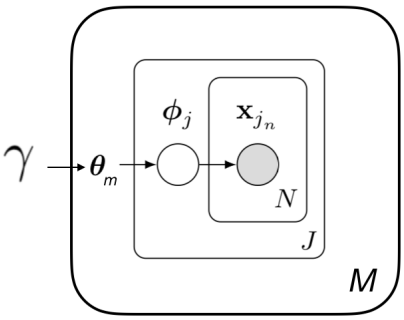

Figure 2: Updated Graphical Model of MAML Inference Clustering

Solving this means solving $p(X|\gamma)$, in which we attempt to marginalize over the two MAML pipelines to find the optimal probabilities of each class belonging to either pipeline so as to maximize our training success. We can see this method unfold with the following sequence of equations. As described in [GRA+18], we can consider MAML as an approximation of the following inference:

$$P(X|\theta) = \prod_j (\int p(X_j|\phi_j)p(\phi_j|\theta)d\phi_j)$$

We assume in our model that each MAML pipeline will approximate this inference. For our task, we want to add one inference step by marginalizing over the possibilities of $p(\theta_m|\gamma)$. With this, our inference task becomes:

$$p(\theta_i|\gamma) \sim Bernoulli(\gamma)$$
$$p(X|\gamma) = p(X|\theta_1) * p(\theta_1|\gamma) + p(X|\theta_2) * p(\theta_2|\gamma)$$
$$p(X|\gamma) = p(X|\theta_1) * p(\theta_1|\gamma) + p(X|\theta_2) * (1 - p(\theta_1|\gamma))$$

Further, we assume that $p(X|\theta_i)$ is approximated by MAML $pipeline_i$. In our work we also create one γ_t parameter for every class or 'task' in our mixed distribution dataset. Thus, for any class t our inference task becomes:

$$p(X|\gamma_t) = pipeline_1 * p(\theta_1|\gamma_t) + pipeline_2 * (1 - p(\theta_2|\gamma_t)$$

Now to complete our inference ideal, we then consider inference over all tasks, and so we take the product over all t.

$$p(X|\gamma) = \prod_t pipeline_1 * p(\theta_1|\gamma_t) + pipeline_2 * (1 - p(\theta_2|\gamma_t)$$

We will try to approximate this inference tasks by initializing γ parameters, sampling tasks, and optimizing a loss function that aims to maximize this likelihood. However, simply training and attempting to optimize this function quickly causes issues since the original two MAML pipelines are not pretrained, and so the model finds it difficult to learn with the usual stochastic gradient descent method how to separate the two datasets. To mitigate this, we consider some simplifications and assumptions to make the problem easier.

To solve a version of this problem, we propose the following algorithm. First, we initialize a separate Bernoulli parameter γ_i for each class in our mixed dataset that corresponds to a probability that $class_i$ belongs to pipeline 1 (and so with probability (1-γ_i) belonging to pipeline 2). We then create a training 'mask' for both pipelines based in which a $class_i$ is only valid for a certain pipeline if its probability of belonging to that pipeline is > .5. (ie: if γ_1 = .7 and γ_{23} = .4, then class 1 will only be trained on pipeline 1 and class 23 only on pipeline 2 in the following training step). We use these masks to then train both pipeline simultaneously for some amount of steps or $metaTrainIterations$. We then update the γ values using a method described in the following sections for a total of $paramUpdateIterations$. Finally, if needed, we then reset the training masks and repeat the pre-training and parameter update process. We now describe more details on each part of the algorithm process.

The following sections will describe parts of the following algorithm. As can be seen, it is broken up into two 'For' loops, one for training of the MAML pipelines, and then one for the γ parameter updates.

Algorithm 1 Inference Clustering using MAML meta-learners

function TRAINCYCLE(γ, $theta$, X, $metaTrainIterations$, $paramUpdateIterations$)
  for $i = 1 \dots metaTrainIterations; i \leftarrow i + 1$ **do**
    $\theta_i, X_i = fn(\gamma)$
    $\theta_i \leftarrow MAML(\theta_i, X_i)$
  end for
  for $i = 1 \dots paramUpdateIterations; i \leftarrow i + 1$ **do**
    $temp \leftarrow MAML(\theta, X)$
    $\gamma \leftarrow paramLoss(temp)$
  end for
end function

4.4 MAML Pipeline Biasing

We first recognize that if both datasets contain a similar amount of classes, then the problem becomes very difficult. This is because if we begin training our pipelines from scratch, both will tend to

converge to a 'mixed' pipeline distribution and will have a difficult time converging to something similar to the separate pipelines trained on the single datasets. However, if we assume that one dataset contains many more classes than the second dataset, then we can use the bias here to attempt to bias one pipeline during initial training towards one dataset versus the other.

Thus, we use the following methodology. We initialize the Bernoulli parameters at the start of training in which some biased amount of the total classes in our mixed dataset (some hyperparameter $b > .5$) have an initial probability of 'belonging' to pipeline 1 of > .5. Our goal is to find a 'sweet-spot' such that dataset 1, of which we are using some significant amount more classes in the pre-training, becomes more sufficiently more adapted to pipeline 1 in comparison to pipeline 2 such that it is distinguishable from dataset 2. What we are hoping for here is that a 'sufficient' number of classes are used to train pipeline 1 in the pre-training phase such that it generalizes better in comparison to pipeline 2, while dataset 2 generalizes in a more comparable manner between pipeline 1 and pipeline 2. More concretely, if we assume there is some s number of classes required before a MAML pipeline will generalize to a sufficient degree. We also assume that $|Dataset_1| = d * |Dataset_2|$ where $|Dataset|$ dictates the number of classes in the dataset. We thus hope that $d * b * |Dataset_2| > |s|$ and $b * |Dataset_2| < |s|$.

Understanding this 'sweet-spot' is still its early stages and requires further study to consider methods to find this for different mixed datasets, but we discuss in a later section the hyperparameters used to find it for our mixed dataset of Omniglot and Fungi.

4.5 Meta Parameter Updates

After we assume that our model has been pre-trained with a particular bias, we then attempt to use this to update the γ parameters. To do so, we now run randomly selected samples from all the classes (instead of just the masked classes) on both pipelines. Our goal is to now exploit the bias of the larger dataset on pipeline 1 to distinguish between the two datasets. To do so, we rely on calculating the following during each run through of both pipelines to compute a parameter loss.

1. The absolute differences between the cross-entropy loss of each D_{test} sample between pipeline1 and pipeline2. Consider a diff for iteration i, on class j. as $DIFF_{ij}$.

$$DIFF_{ij} = P1Loss(D_{test_{ij}}) - P2Loss(D_{test_{ij}})$$

2. The minimum absolute loss difference between $pipeline1$ and $pipeline2$ on any iteration i. Call this $minDIFF_i$.

$$minDIFF_i = min(DIFF_{i1}, DIFF_{i2})$$ (assuming n-way = 2)

3. A sample mean amongst all iterations of the absolute difference between the minDIFF of any iteration and all $DIFF_{ij}$ of that iteration. Call this the $MinSampleMean$.

$$minSampleMean = \frac{1}{totalSamples} \sum_i \sum_j minDIFF_i - DIFF_{ij}$$

4. We then compute a sample-loss-diff by taking a signed difference between the min-sample-mean and the current $Diff_{ij}$ for class j on iteration i. For each sample, we will call this the $LossDiff$

$$LossDIFF_{ij} = minSampleMean - DIFF_{ij}$$

5. We use this to compute the loss similarly to marginalizing over a Bernoulli variable. For any particular class t, we call this the $ParamLoss(t)$

$$ParamLoss(t) = p(\theta_1|\gamma_t) * (LossDIFF_{ij}) + p(\theta_2|\gamma_t) * (-LossDIFF_{ij})$$

We then update the γ parameters using the following update rule with a new hyperparameter the $metaUpdateRate$.

$ParamLoss(t) = ParamLoss(t) - metaUpdateRate * \nabla(ParamLoss(t))$

5 Experiments and Results

We run here two sections of experiments. In the first section we attempt to show how training the Omniglot and Fungi results in image classification performance both with the baseline MAML models, with the K-Means pre-separation, and with the separation from the Inference model. In the second section we then sweep a few hyperparameters of the Inference model to see how changing the initial Bernoulli bias and changing the initial number of pre-training steps affects the ability to classify each class in the mixed dataset.

As a note, for these results we train the models on nearly the full omniglot dataset (with 1600/1623 classes) and on a shortened Fungi dataset (with only the first 400/1393 classes). This is to make our tests consistent with the environment required to test the Inference algorithm.

5.1 MAML Pipeline Model Comparisons

In the first section of results, we compare the meta-validation accuracies during training of MAML models that are trained with the following scenarios. First, we train the models individually using only the Omniglot and Fungi datasets separately. We then trained models on two separate masked datasets that were pre-clustered by K-Means. Finally, we train the models with masked datasets that were pre-clustered by Inference method.

Model Parameter	**Value**
n-way	5
k-shot	1
inner-update-lr	.4
number-inner-updates	1
meta-batch-size	8
meta-train-iterations	5000
learn-inner-learn-rate	True

Table 1: General Hyperparameters for MAML Testing

The K-Means and Inference results were each trained with two clustered datasets which reached a separation accuracy as shown in this table. The hyperparameters used to train the inference model are in the table shown in the following hyperparameter section.

Model	**Omni Recovery**	**Fungi Recovery**
K-Means	.91	.87
Inference	.73	.64

Table 2: Separation Accuracy of Methods

In these plots, we compare the training of the different models by tracking meta-validation accuracy over iterations. In the first plot, we compare the training of MAML with the Omniglot set, the Fungi set and with a dataset with a random mixture of the two. The next two plots compare the pure dataset (Omniglot and Fungi) with a clustered dataset separated by either the K-Means algorithm or the Inference algorithm we propose. Interestingly enough, we see that the mixed dataset has similar meta-validation accuracies to the individual datasets. Our true tests in understanding the limits of the task-distributions will be rather when testing mixed models with individual datasets, which we will show in some upcoming accuracy tables.

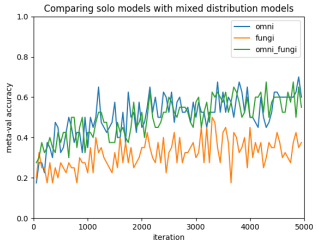

Figure 3: Comparing Validation Accuracy Models

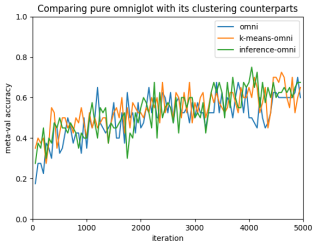

Figure 4: Comparing Omniglot model with Omniglot separated clusters

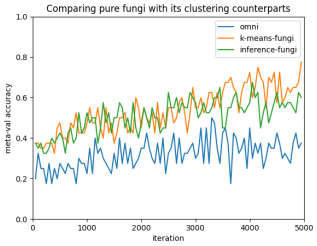

Figure 5: Comparing Fungi model with Fungi separated clusters

In the following table, we trace accuracies of several different type of trained models at meta-test time. We consider this as the most accurate method to get a sense to how well any trained model on a single task-distribution is able to correlate to another dataset. An interesting starting point is that testing both Omni and Fungi datasets on the Random Mixed model performs worse than training the models individually.

Model	**Omni Test Accuracy**	**Fungi Test Accuracy**
Omni only	.61	.20
Fungi only	.21	.38
Random Mixed	.52	.20
K-Means Omni	.56	.31
K-Means Fungi	.56	.24
Inference Omni	.59	.30
Inference Fungi	.52	.24

Table 3: Meta-Test accuracies of different models

5.2 Mixed Dataset Clustering Results

In this section, we see how the clustering results upon the Mixed Task Distribution dataset varies as we sweep through some of the hyperparameters that we describe in the following section. From we can clearly see that training the model as we have it is quite sensitive and will requires work to find good results. All of these are run a mixed Omniglot and Fungi dataset in which the Omniglot outnumbers the Fungi dataset by number of classes by a factor of 4. Though we see some degree of equilibrium in the plots below, we believe that these params are likely very dataset dependent as the model currently is. Part of the future work will be to find a less volatile system that accomplishes the same task.

We summarize the hyperparameters for the MAML model used to create these results in the following section. In the first set of plots, we compare the clustering of both Omni and Fungi classes when we sweep through the number of initial meta-train step iterations while keeping the initial $pipeline1$ bias at .75. In the second set of plots, we compare results when we always train for 200 iterations and sweep through different initial bias. From these results, we came to find that our current optimal set of hyperparameters are an initial bias of .65 and a pre-training number of 200 iterations.

(Note that in the following plots, the true clustering accuracy of Fungi will be the value in the x-axis subtracted from 1, which will the probability that a Fungi class will be sent to pipeline2. Also note that our goal in these plots is generally for the line in the Omni case to stray as high as possible while the line in the Fungi case strays as low as possible)

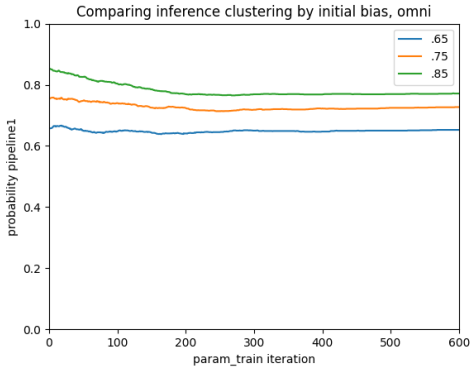

Figure 6: Clustering Accuracy of Omni while sweeping initial Bias

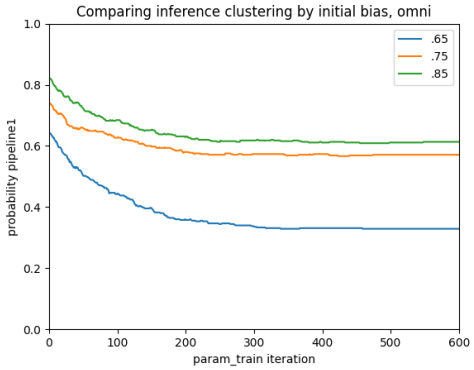

Figure 7: Clustering Accuracy of Fungi while sweeping initial Bias

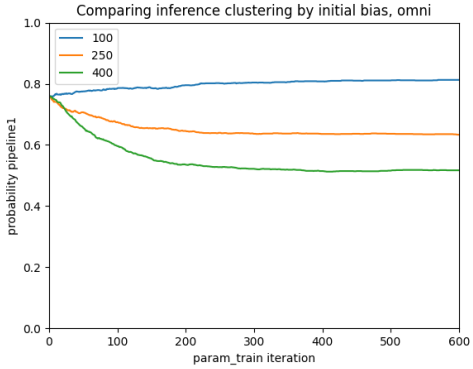

Figure 8: Clustering Accuracy of Omni while sweeping meta-train iterations

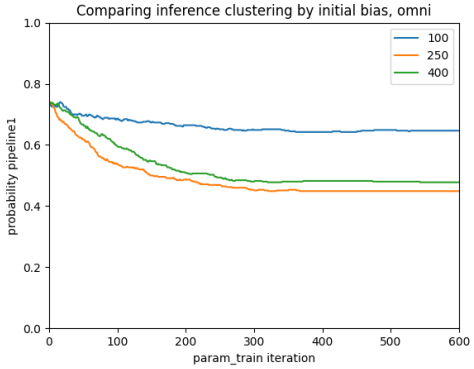

Figure 9: Clustering Accuracy of Fungi while sweeping meta-train iterations

6 Error Analysis and Hyperparameter Tuning

Though we were able to find a suitable set of parameters for our model in which to see some results for the mixed Omniglot and Fungi dataset, in trying the model for different mixed datasets it becomes clear that finding the right parameters to run the model can be quite finicky depending on the dataset being trained. We consider the following table to be a list of hyperparameters.

Model Parameter	**Value**
n-way	2
k-shot	1
inner-update-lr	.4
number-inner-updates	1
meta-batch-size	8
meta-train-iterations	200
learn-inner-learn-rate	True

Table 4: MAML Hyperparameters for Inference

Model Parameter	Value
Train-Cycle Iterations	200
Param-Update Iterations	600
Number Total Cycles	1
Initialization Bias	.65
MetaUpdateRate	1e6
Dataset Size Bias Factor	4

Table 5: Added Hyperparameters for MAML Inference

Here, Train-Cycle Iterations refers to the number of pre-trained iterations of the MAML pipelines before we begin the Param Update step. Param-Update Iterations is the number of iterations we update the γ params. Number of Total Cycles is the number of times we train/update params, then repeat. Finally, the Dataset Size Bias Factor describes the multiple of which our pipeline1 biased dataset outnumbers the second dataset by number of classes. (In this case, it is a factor of 4).

In tuning the model, the strategy was generally to try parameters until we empirically found that the minimum pipeline difference sample mean (as described in the previous section) was significantly larger for dataset 1 rather than dataset 2. However, even in these instances, it wasn't always the case that our loss function forced the parameters in the direction we were hoping. In these cases, we saw the problem of outliers dictating the averaging when the intermediate samples were far more moderate. This issue increased the difficulty of finding the right set of parameters in which to train the model, and perhaps leads us to try to think of a more robust loss function for updating the parameters.

7 Conclusions and Further Work

By proposing an algorithm that use only MAML pipelines to cluster mixed task-distribution datasets, we open a door to consider in what ways we can leverage the natural properties of meta-learners to gain insights to our data in an unsupervised way. Our results show that it is possible to use the natural higher-level 'task-distribution' of these datasets, and the natural properties of MAML pipelines to distinguish classes of data at a higher level.

For future work, we will now want to experiment and understand how well our model might work on different mixed datasets. From there, assuming we get different degrees of clustering success on different datasets, we want to try to play around with implementing shared layers between the two meta-learning pipelines such that the meta-test accuracies of both individual datasets increases. This considers a more general theory of how algorithms such as this one reveal information about the MAML model itself, and to what degree its own behavior can tell us about its clustering limitations and to what degree of true philosophical difference exists between different 'task-distributions.'

Codebase

https://github.com/sjmickiemouse/cs330_project.git

Contributions

References

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[Fin+20] Chelsea Finn, Karol Hausman, et al. "CS 330: Deep Multi-Task and Meta Learning". In: (2020). URL: https://cs330.stanford.edu/.

[FXL18] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic Model-Agnostic Meta-Learning". In: *CoRR* abs/1806.02817 (2018). arXiv: 1806.02817. URL: http://arxiv.org/abs/1806.02817.

[Gar+18] Marta Garnelo et al. "Conditional Neural Processes". In: *CoRR* abs/1807.01613 (2018). arXiv: 1807.01613. URL: http://arxiv.org/abs/1807.01613.

[Gra+18] Erin Grant et al. "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes". In: *CoRR* abs/1801.08930 (2018). arXiv: 1801.08930. URL: http://arxiv.org/abs/1801.08930.

[HLF18] Kyle Hsu, Sergey Levine, and Chelsea Finn. "Unsupervised Learning via Meta-Learning". In: *CoRR* abs/1810.02334 (2018). arXiv: 1810.02334. URL: http://arxiv.org/abs/1810.02334.

[RB19] Sachin Ravi and Alex Beatson. "Amortized Bayesian Meta-Learning ". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=rkgpy3C5tX.

[San+16] Adam Santoro et al. "Meta-Learning with Memory-Augmented Neural Networks". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1842–1850. URL: http://proceedings.mlr.press/v48/santoro16.html.

[SSZ17] Jake Snell, Kevin Swersky, and Richard S. Zemel. "Prototypical Networks for Few-shot Learning". In: *CoRR* abs/1703.05175 (2017). arXiv: 1703.05175. URL: http://arxiv.org/abs/1703.05175.

[Tri+19] Eleni Triantafillou et al. "Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples". In: *CoRR* abs/1903.03096 (2019). arXiv: 1903.03096. URL: http://arxiv.org/abs/1903.03096.

[Yao+19] Huaxiu Yao et al. "Hierarchically Structured Meta-learning". In: *CoRR* abs/1905.05301 (2019). arXiv: 1905.05301. URL: http://arxiv.org/abs/1905.05301.