Semi-Supervised Task Construction for Meta-Training
CS330 Project Report. SUNet ID: arkanath

Abstract
Meta Learning, on a fundamental level, can be described as learning to optimize efficiently for new tasks. In the most common setup, there is a distribution of tasks (classification tasks in our case) and the meta learning algorithm learns from the train set of tasks in order to perform well on unseen tasks. This meta-training process is typically performed on a labeled dataset, however, recent work has shown that unlabeled datasets can also be useful for meta-training. CACTUs [3] uses unsupervised partitioning for task construction during meta-training. However, in many real world scenarios we also have access to some amount of labeled data for training, which is the motivation behind this work. We extend the setup proposed in CACTUs and change the problem to arrive at a better partitioning, using a small amount of labeled data.

The contributions of this project are two-fold: first, we formally define the problem with two variants: Disjoint and Superset. These two variants relate to the overlap between the underlying classes of the unlabeled and labeled data. The Disjoint variant is inspired by the scenario where we have labeled data for a given set of classes and unlabeled data for classes outside of the labeled distribution. The Superset variant considers a problem where we label a subset of the overall dataset. This will also be relevant in an Active Learning setup, which we consider.

Second, we propose multiple directions for tackling this problem, and we perform insightful studies along those directions. Our first approach merges the labeled tasks with the clusters obtained from the unlabeled data during training. We show that only 10% labeled data results in a significant boost in test performance. We also study the performance over more levels of label coverage. For the second approach, we hold out labeled tasks for validation. We perform a study over the number of clusters and show a consistent trend of increasing test performance. The first two approaches work well under the Disjoint assumption. For the third direction, we propose seeding the cluster centroids based on the labeled data. We experiment with different ideas, however, we see a consistent drop in performance with these approaches - which makes us believe that the tasks don’t exhibit a perfect structure in the embedding feature space. As a fourth approach, we exclude clusters in order to enforce constraints derived from the labeled data. With a similar outcome, this also doesn’t help improve the performance over the unsupervised baseline. We also experiment with the k-medoids clustering algorithm in order to enforce data points as centroids, but k-medoids does not perform as well as k-means in this setup. Inspired by these studies, we deduce that solely altering the task construction favorably using labeled data is not an easy problem in the Superset variant. We note here that the scope can inherently be limited if the embedding feature space can not serve well for identifying the task distribution in a semi-supervised fashion. To investigate further, we study the effect of training only with the smallest or the largest clusters. We observe that the larger clusters are more useful when compared to the smaller clusters, when keeping the same number of clusters. This could be a result of more data or because of more coherent clusters. However, we also note that training only on the larger or smaller clusters does not result in better performance.
Introduction
With the advent of high computational power and large datasets, Machine Learning is rapidly gaining usage in more areas. However, both of these resources have associated costs and are not easily accessible in many real world scenarios. In the real world, we often have to deal with tasks where we don’t have enough data to learn from. In the case of few-shot classification, which is the focus for this report, we have to classify data after learning from only a few examples for each class. The training process (a.k.a. meta-training) uses a set of labeled tasks coming from the same task distribution. There are a range of algorithms that have been shown to work well in such a setup. MAML [1] optimizes the weight initialization for a neural network in order to make it easy to fine-tune for new tasks. ProtoNets [2] learn prototype representations which perform better when fetching the nearest class for a query. SNAIL [11] uses sequence modeling to learn from previous experience. These algorithms are being used successfully for many real world applications. However, in many such scenarios, while we only have access to a few labeled examples for the tasks being observed at serving, we often have access to larger unlabeled datasets that come from similar task distributions at the time of training.

Recent work [3, 4] has shown that meta-training can be performed exclusively using unlabeled datasets, and still learn something meaningful that performs well. One such work, CACTUs [3], shows that we can use an unsupervised partitioning algorithm such as k-means clustering to artificially construct meta-training tasks, and use a standard Meta Learning algorithm like MAML or ProtoNets with them. This approach shows impressive performance, which in some cases even gets close to what we’d achieve learning from a fully labeled dataset.

In this work, we consider the scenario in which we have access to a small amount of labeled dataset, along with a larger, unlabeled, dataset for training. This scenario is also popularly known as semi-supervised learning, which is extensively studied and used for the traditional supervised learning problems, but not as much for meta learning. In the meta learning framework, we also have an additional distribution of tasks, aside from the distribution of data. But there’s a natural parallel that can be drawn here as well. CACTUs shows that unlabeled data can be used to learn a useful prior for meta-training, simply by constructing meta-training tasks. Our work should be considered a natural extension to CACTUs as we use the same setup. In this work, we build on CACTUs results and question whether a small amount of labeled data can help in this partition construction process. While there has been some work explored along Semi-supervised Meta Learning [5, 6], those few approaches tend to alter the meta-learning algorithms fundamentally in some way whereas our focus is solely on the task construction process. The contributions of this project are two-fold:

1. We formulate the problem with two variants: Disjoint and Superset, and provide natural motivations for each.
2. We propose multiple approaches for tackling this problem and perform insightful studies along those directions.

Semi-Supervised Task Construction
In this section, we describe our problem setting and the experiment setup.
Our work is an extension to CACTUs [3]. CACTUs performs unsupervised meta-training by using unsupervised partitioning to automatically construct tasks for training. CACTUs assumes that the downstream tasks are M-way classification tasks and the goal is to learn an accurate classifier using K labeled data points from each of the M classes, where K is relatively small. The unsupervised meta-training phase aligns with the unsupervised learning problem in that it involves no access to information about the downstream tasks, other than the fact that they are M-way classification tasks, for variable M upper-bounded by N. The upper bound N is assumed to be known during unsupervised meta-training, but otherwise, the values of M and K are not known a priori. CACTUs uses k-means and hyperplanes as partitioning methods. The unsupervised partitioning is performed using two different pre-trained embeddings as features (ACAI and BiGAN), and they experiment with 4 different datasets (Omniglot, miniImageNet, CelebA, and MNIST). We limit our experiments to ACAI and Omniglot only, in the interest of time. They also repeat the partitioning multiple times in order to obtain a variety of partitions, however, in the interest of time, we limit to a single partition as we observed the difference in performance was not very significant in order to affect the outcome of our studies. The dataset is split into train, val and test sets. For the Omniglot dataset (the focus of our studies), the classes are split disjointly so the underlying classes are not shared at meta-train time.

Problem Formulation

While CACTUs works in a completely unsupervised setup, we extend that to explore the semi-supervised setting. We'll use the same setup but include a small amount of labeled data available to us at the time of training. While doing that, we consider two variants in terms of how the underlying classes between the labeled and unlabeled data are related.

Disjoint

In the Disjoint setup, we assume that the unlabeled and the labeled data do not share classes, but they do come from the same underlying task distribution. This is inspired by the real world scenario where the labeled data is collected separately from the unlabeled data. Examples could include scenarios of collecting robotic manipulations or self-driving data which will not be found in the unlabeled data, which would be collected with a different routine.

Superset

We consider another use case which may be more common, especially for the supervised classification problem setting. In this Superset variant, we explore the setting where the underlying classes of the unlabeled data is a superset of the classes in the labeled data. This is a common outcome of labeling a small subset of a larger unlabeled dataset. Going further, we consider two possibilities here as well:

1. The labeled data is a random sample of the full dataset.
2. We have the freedom of choosing what to label from the full (unlabeled) dataset.

The second scenario can be considered an instance of Active Learning in this case.
Studies

Experiment Setup
We restrict our experiments to the Omniglot dataset. Omniglot is a widely-used few-shot classification dataset, containing 1623 character classes, with 20 examples each. We follow the same setup as CACTUs, splitting the dataset into 1100, 100 and 423 classes for the train, validation and test splits respectively. We also restrict our studies to ACAI embeddings as features for semi-supervised partitioning. We choose MAML as the meta-learning algorithm of choice for our studies. After partitioning, we follow the same setup as CACTUs for task construction and train MAML for 300,000 steps for all experiments for 5-way 1-shot classification. In the CACTUs workflow, we restrict the number of clustering runs to one when generating partitions, in the interest of time. We observed that the difference in performance was small and should not affect the outcome of our studies. For most of our experiments, we use 10% of the data as labeled, i.e. 2200 examples out of 22000 (=1100 * 20) examples used during training.

In the test setting, we evaluate the accuracy for both 5-way 1-shot and 5-way 5-shot classification. We report the final accuracy after 50 gradient updates in the tables, and report the accuracy for the first 5 updates in the charts.

Disjoint Setting
For the Disjoint setting we propose two different directions for utilizing the labels.

Merging labeled tasks with clusters from unlabeled data
In this approach, we union the labeled task partition with the partition obtained by clustering (k-means) to obtain the final partition for task construction. We study the variance in the test performance as we increase the % labeled classes. While doing this, we fix the number of clusters as 500 for the unlabeled data. The results are reported in the charts and table below.
We note that this approach works well and that we see an increase in performance from 0.856 to 0.912 with having only 10% labeled data in the 5-way 5-shot test scenario, bridging the gap by almost half for what we can achieve by having 100% labeled data (0.987). We also compare this with an experiment where we train by using only 10% labeled data as tasks and we see a performance of 0.883, more than that of training with all the unlabeled data. For the 5-way 1-shot test scenario, we see a more gradual increase in performance as we gain more label coverage.

Tuning parameters using labeled tasks for meta-validation

In the original CACTUs work, they do not tune any parameters using the labeled meta-validation data, so as to achieve a completely unsupervised training framework. However, in our case, we can choose to tune the unsupervised task construction parameters by holding out labeled tasks for meta-validation. For our study here, we chose to tune the number of clusters which was fixed as 500 in CACTUs. The results are reported in the table below.

<table>
<thead>
<tr>
<th>No. of clusters</th>
<th>(5 way, 5 shot)</th>
<th>(5 way, 1 shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.804</td>
<td>0.610</td>
</tr>
<tr>
<td>250</td>
<td>0.841</td>
<td>0.650</td>
</tr>
<tr>
<td>500</td>
<td>0.856</td>
<td>0.662</td>
</tr>
<tr>
<td>750</td>
<td>0.861</td>
<td>0.673</td>
</tr>
<tr>
<td>1000</td>
<td><strong>0.864</strong></td>
<td><strong>0.681</strong></td>
</tr>
<tr>
<td>1100</td>
<td><strong>0.864</strong></td>
<td>0.675</td>
</tr>
<tr>
<td>1500</td>
<td>0.861</td>
<td>0.677</td>
</tr>
</tbody>
</table>

We see increased performance as we increase the number of clusters. Interestingly, the best performance is reached when the number of clusters is close to the original number of tasks (underlying classes).

Note that we studied the two approaches above separately but we can also mix both together.

Superset Setting

For the Superset setting as well, we propose two different directions.
Seeding cluster centroids using the labeled data

For this first direction, we intend to initialize the cluster centroids favorably by leveraging the labeled data. Basu et al. [7] study semi-supervised clustering using seeding in which they show what they call “seeded k-means clustering” can be effective for maintaining flexible constraints. The idea is simple: instead of random initialization, they initialize the cluster centroids as the mean of examples from a label.

We begin by studying the use case where we have a random sample of the data labeled. For our training data, we have 1100 classes and 20 examples. When we have 10% of our examples labeled randomly, we arrive at discovering around ~970 unique labels out of the 1100, with each class having an average of ~2.3 examples. For meta-training task construction, we require 6 examples in each class, but only 3 of the classes we discover have more than 6 examples. Therefore, clearly, a 10% random sample is not enough for meta-training with the native approach.

When using seeded k-means clustering, we explore three approaches. For the first approach, we use 1100 as the number of clusters, building on the assumption that we can estimate the underlying number of classes. We initialize the ~970 centroids using the mean of examples found during labeling, while the rest of the cluster centroids are initialized randomly from the unlabeled examples. As a second approach, we limit the clusters to the labels discovered in the random sample. For a third approach, we perform hierarchical clustering using the minimum of the distances for linkage, to arrive at a smaller number of 250 cluster centroids. With this way, we merge together the labels which are similar in the embedding feature space. We perform k-means clustering with these derived 250 cluster centroids as seeds.

<table>
<thead>
<tr>
<th></th>
<th>(5 way, 5 shot)</th>
<th>(5 way, 1 shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised (1100 clusters)</td>
<td>0.864</td>
<td>0.675</td>
</tr>
<tr>
<td>1100_clusters_mean_seeded</td>
<td>0.672</td>
<td>0.368</td>
</tr>
<tr>
<td>clusters_mean_seeded</td>
<td>0.791</td>
<td>0.507</td>
</tr>
<tr>
<td>clusters_any_seeded</td>
<td>0.798</td>
<td>0.491</td>
</tr>
<tr>
<td>HAC_seeded_250_clusters</td>
<td>0.842</td>
<td>0.642</td>
</tr>
</tbody>
</table>

We notice poor performance as we try these seeding approaches. The closest performance comes from seeding with HAC merged labels, which is still not able to exceed the performance achieved by the unsupervised baseline, which uses random seeding. This behavior suggests that explicit seeding is not a right approach in this setting. In the next section, we explore semi-supervised clustering by using constraints derived from the labeled data.
Selective sampling of clusters by using constraints derived from the labeled data

With this direction, we intend to remove clusters that may not align with the underlying class distribution, in the hopes of arriving at a more suitable partition for task construction. Using known constraints to perform semi-supervised clustering has been explored before. One such algorithm, called the COP-KMEANS algorithm [8], does so by enforcing the known constraints in the assignment stage of the k-means algorithm. The COP-KMEANS algorithm works with two types of constraints: must-link and cannot-link pairs. For this study, we remove clusters that invalidate constraints to improve over the baseline.

For the first approach, we continue in the random label sample setting, and therefore deriving must-link and cannot-link pairs from the labels obtained from a random sample of the overall dataset. Once we have the clusters obtained in a completely unsupervised setting, we perform these two operations to invalidate clusters:

1. Invalidate any cluster that contains two examples labeled differently.
2. For every pair of examples belonging to different clusters but coming from the same label in the random sample, invalidate both corresponding clusters.

For the next two approaches, we explore the freedom of choosing what to label from the overall dataset. We propose two approaches here, both of which operate after the unsupervised clustering has been performed. The first approach keeps coherent clusters by sampling two examples from each cluster at random, and invalidating the clusters for which the labels of the two examples don’t align. The second approach removes incoherent clusters, by labeling all examples from the smallest clusters. We label as many clusters as possible within the labeling quota (In our setting of 10% labeled examples, we were able to label ~25% smallest clusters). We then invalidate any cluster which doesn’t have at least half of its examples belonging to the same label. We report the results of these three approaches in the following table.

<table>
<thead>
<tr>
<th></th>
<th>(5 way, 5 shot)</th>
<th>(5 way, 1 shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised (1100 clusters)</td>
<td>0.864</td>
<td>0.675</td>
</tr>
<tr>
<td>Enforce constraints</td>
<td>0.749</td>
<td>0.554</td>
</tr>
<tr>
<td>Keep coherent clusters</td>
<td>0.691</td>
<td>0.496</td>
</tr>
<tr>
<td>Remove incoherent clusters</td>
<td>0.723</td>
<td>0.441</td>
</tr>
</tbody>
</table>

With these approaches too, we fail to arrive at an improved performance over the unsupervised baseline. To investigate further, we also study the effect of removal of clusters based on size.
We observe here that the larger clusters are more effective on their own than the smaller clusters, however, removal of either the smallest or largest clusters don’t help improve the performance over the baseline. The larger clusters could be more effective due to more data or potentially because of more coherency within the clusters.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Test Accuracy (5-way 5-shot)</th>
<th>Test Accuracy (5-way 1-shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smallest 50% clusters</td>
<td>0.689</td>
<td>0.456</td>
</tr>
<tr>
<td>Largest 10% clusters</td>
<td>0.707</td>
<td>0.516</td>
</tr>
<tr>
<td>Largest 25% clusters</td>
<td>0.814</td>
<td>0.619</td>
</tr>
<tr>
<td>Largest 50% clusters</td>
<td>0.848</td>
<td>0.652</td>
</tr>
</tbody>
</table>

**Conclusion**

We formulate the semi-supervised task construction for meta-training in two variants, Disjoint and Superset, both of which are inspired by real world use cases. For the Disjoint setting, we show that merging labeled tasks and unsupervised clustering obtained from the unlabeled data results in an effective partition for meta-training. We also show that tuning the parameters for clustering like the number of clusters can be effective using a held out labeled dataset for meta-validation. For the Superset setting, we propose two different directions. The first direction uses seeded k-means in order to favorably influence the unsupervised partitioning. The second direction invalidates clusters from the unsupervised clustering based on constraints derived from labeled data. For both directions, we study multiple approaches, none of which result in an improved performance over the unsupervised baseline. We explain this by inherent limitations potentially imposed by the embedding feature space for identifying the underlying task distribution. To investigate further, we show that neither the smallest or largest clusters contribute negatively in the unsupervised partitioning for meta-training. However, we note that the larger clusters were more effective, due to more data or potentially because of more coherency within the clusters. Finally, one should note that these studies are based on our experiment setup, and the outcome may be different with another dataset or embedding feature space, even though we have tried to keep our experiments generic in nature.
References


