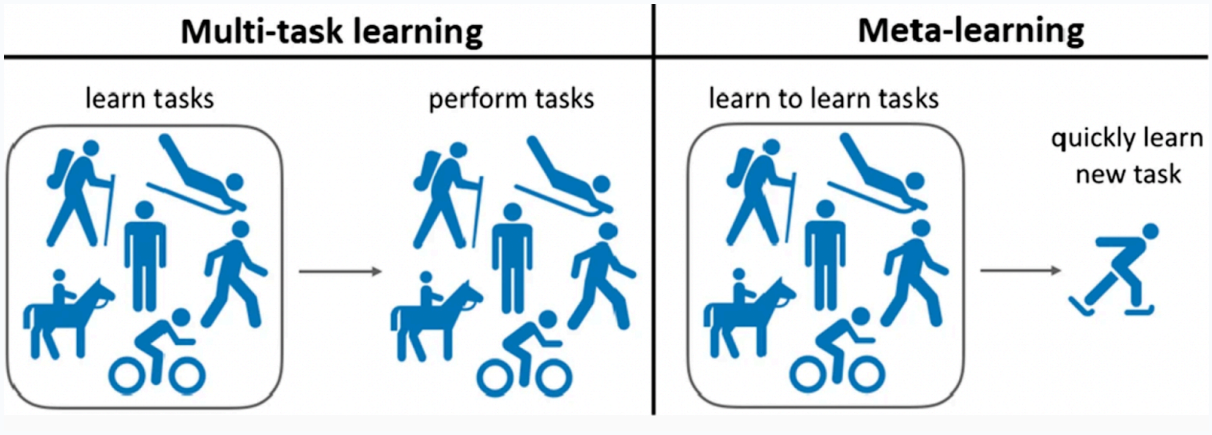
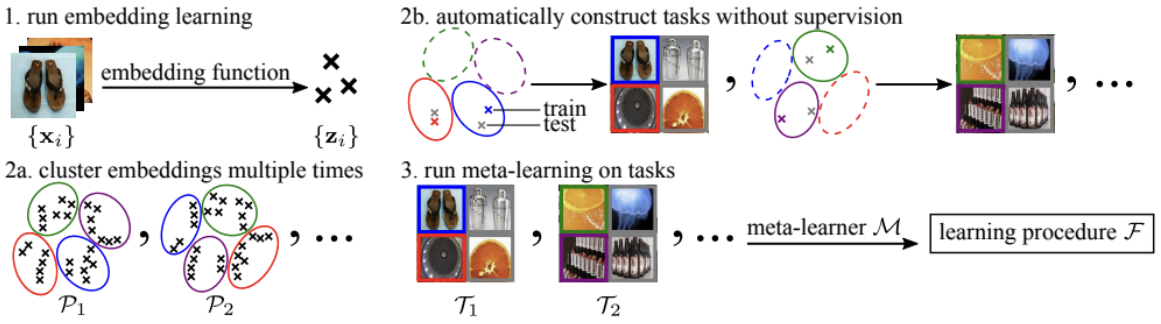
Meta-learning: how does it work?

The application of machine learning algorithms or optimization and training of machine learning models is known as meta-learning. When an AI system uses meta-learning, it can apply the concepts it learned for one task to various complex tasks. Meta-learning can be implemented in multiple ways depending on the type of work and task at hand. As a result of meta-learning, one network's parameters are transferred to other networks' parameters. As far as meta-learning is concerned, there are two phases. The meta-learning model is typically trained after the base model has completed multiple training steps. Following the forward, backward, and optimization phases that train the base model, forward training is performed on the optimization model. After calculating the meta loss, the gradients of each meta parameter are calculated. Using an original model's forward training pass and then aggregating the previous losses is one method for determining the meta loss.
There are usually hundreds and thousands of meta-learning parameters in many deep-learning models. However, creating a meta-learner for every new set of parameters would be an expensive investment. Thus, coordinate sharing can be used to resolve this problem.



It has always been a story of abstraction in machine learning. Earlier researchers spent their days creating tasks manually for machine learning models, but with unsupervised meta-learning, algorithms now propose their task distribution. Unsupervised meta-learning reduces the amount of human intervention needed to solve tasks. Let's look at a few examples of machine learning algorithms that use human supervision to find patterns and extract knowledge. A common ML scenario is a regression, where a human provides a label Y for a few examples. The goal is to return predictions that correctly assign the labels to these examples. Reinforcement learning (RL) is another example where the input is a dataset, and the output is a function that performs well on the dataset. RL algorithms use many tools, such as kernel, tabular, and deep neural networks. Unsupervised learning aims to acquire representations from unlabeled data that can be used to learn downstream tasks more effectively from modest amounts of labeled data. Several prior unsupervised learning works have attempted to accomplish this by developing proxy objectives based on reconstruction, disentanglement, prediction, and other metrics. As an alternative, we develop an unsupervised meta-learning method that explicitly optimizes the ability to learn from small amounts of data. To accomplish this, we automatically construct tasks from unlabeled data and then run meta-learning on them. As a surprising finding, we find that relatively simple

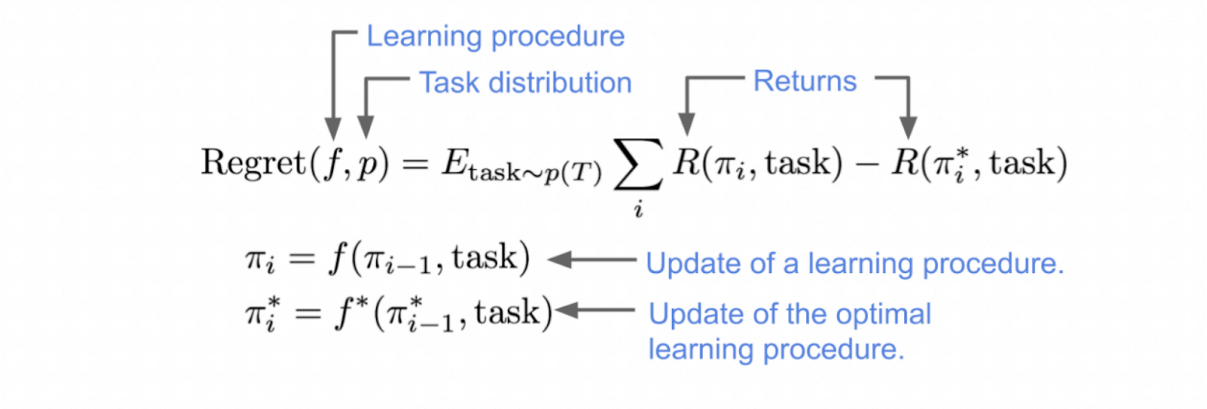
task construction mechanisms, such as cluster embeddings, are effective when integrated with meta-learning. Based on our experiments across four image datasets, we have demonstrated that our unsupervised meta-learning approach produces a learning algorithm without any labeled data that applies to a variety of downstream classification tasks, improving on the embeddings learned by four previous unsupervised learning methods.



In the context of unsupervised meta-learning, we can imagine a meta-learner that can achieve the worst-case regret in all possible task distributions within an environment. The unsupervised meta-RL algorithm performs significantly better than learning from scratch and is competitive with supervised meta-RL approaches on benchmark tasks. It is believed that meta-learning algorithms with memory may perform best when the task proposal mechanism is different. One of the most promising techniques for achieving artificial general intelligence is meta-learning. As a result of unsupervised meta-learning, this goal is one step closer, allowing humans to solve tasks with less supervision. If we are trying to solve many tasks, it is complicated for the machine learning practitioners to translate all of them manually; hence it is necessary to make the process automotive. The algorithms that perform this optimization automatically are called meta-learning algorithms. It is important to note that meta-learning algorithms consider not only one task but a distribution over many tasks when evaluating a set of knobs. There may be a distribution over supervised learning tasks that includes learning dog detectors, cat detectors, and bird detectors, for instance. The task distribution in tasks is interrelated, information from one task may help solve other tasks more efficiently, but a knob setting that works well on one task may not be the same for others as the knob setting depends on the distribution of tasks. The task distribution for meta-learning cannot be done without no prior knowledge in mind.

Some Calculations

Defining an optimal meta-learner for the case when the distribution of tasks is unknown is the first step. It is generally accepted that an optimal meta-learner achieves the greatest average reward across the distribution of functions. We will compare the expected reward for a learning procedure f to that of the best learning procedure f∗ , defining the *regret* of f on a task distribution p more accurately.



$$\min_f \max_p \quad \text{Regret}(f, p).$$

$$\mathcal{I}(s, z) = \mathcal{H}(s) - \mathcal{H}(s|z).$$

An optimal unsupervised meta-learner can be defined as a meta-learner that returns the minimum worst-case regret among all the task distributions encountered. Datasets are converted into function approximators using learning procedures. One can tune the learning procedures by optimizing the learning procedures to solve a distribution of tasks. Designing the task distribution manually is challenging, so recent research suggests that learning algorithms can optimize knobs using unlabeled data. It provides bounds on the regret we might incur in the worst-case scenario from such an unsupervised meta-learner. The optimal distribution for an unsupervised meta-learner is uniform over all possible tasks under some restrictions on the family of tasks encountered at test time.

The first marginal entropy term is maximized when all possible tasks have a uniform distribution. For every conditional entropy term, we ensure consistency by Z, and the resulting distribution of S is narrow. Using the notion of min-max optimality, we can construct unsupervised task distributions in an environment by optimizing mutual information. When the test time task distribution is capable of being chosen adversarially, the algorithm must make sure it is uniformly good for all the tasks that could be encountered. To illustrate, if rewards at test time were limited to goal-reaching tasks, regret for reaching a target during test time is inversely related to training-time sampling of the target. An adversary may propose a task distribution solely for reaching one of the goals g if it has a lower density than the others, causing the learning procedure to incur a higher regret. Uniform distribution over goals can be used to find an optimal unsupervised meta-learner.

Applications of Unsupervised Meta-Learning

In what ways could meta-learning be beneficial, or are embeddings already sufficient for downstream supervised learning? We investigate this by running MAML and ProtoNets on tasks generated via CACTUs (CACTUs-MAML, CACTUs-ProtoNets). In this study, we compare five alternative algorithms, four of which incorporate supervised learning. ii) Embedding knn-nearest neighbors first infers the embeddings of the downstream task

images. In the embedding space, it predicts the plurality vote of the labels of the KNN training images closest to the query embedding. It employs a network with a 128-unit hidden layer and well-tuned dropout instead of an embedded multilayer perceptron (Srivastava et al., 2014). To isolate the effect of meta-learning on images, we also compare it to embedding cluster matching, which involves labeling clusters with training data. In cases where a query data point maps to an unlabeled cluster, the closest labeled cluster is used. v) For each evaluation task, we use gradient descent to train a model from standard random network initialization using the MAML architecture.

By using simple embeddings in tasks, meta-learning improves the utility of these representations in learning downstream, human-specified tasks. For more detailed information please refer to the research here.