Learning to Learn

Charlie Wu

May 2019

1 Introduction

Learning to learn can be described simply as *meta-learning*, or the process of being able to design and train successful optimization algorithms in a similar way that features are currently learned in vanilla machine learning models. In recent years of machine learning research, the process of learning features in models has become very well refined and effective, in comparison to the previous process of utilizing features that have been created manually.[4] While various researchers differ in exact *meta-learning* terminology, we will utilize definitions described by DeepMind researchers in their seminal paper published in 2016.[2] This process of using optimization algorithms created by humans, while very accurate when utilized to solve specific problems, are not able to quickly train models that are generalized over wide sets of similar problems, or families of related tasks. This severely restricts adaptability when training models involving many related tasks, and creates issues in training effectively when new training data is limited or when training on new or novel examples.

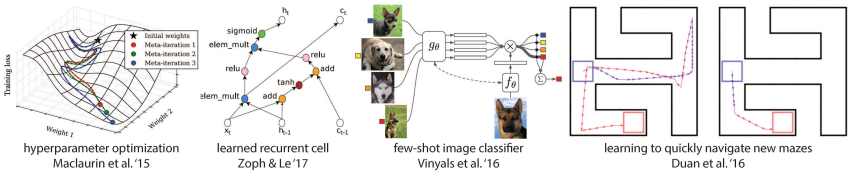

Figure 1: Various examples of meta-learning in action [3]

2 Background

The general goal of *meta-learning* systems is to obtain a learning algorithm that functions well when applied to training models for an entire class of problems, rather than training a singular model on one specific problem from the set of problems in a class. Such a learning algorithm should be able to train new models both quickly and accurately compared to when a learning algorithm defined by hand is used. The creation of aforementioned learning algorithm is

viewed as a learning problem in and of itself. Here, we will focus on the definition of *meta-learning* involving training optimization algorithms for training models. In this lecture, we will refer to the vanilla model being trained as the *base-learner* and the high-level model that is training to create an optimization algorithm for the *base-learner* as the *meta-learner*. In any case, the *meta-learner* will train and continually try to improve efficiency and minimize loss as various *base-learners* train on their tasks over a family of tasks. Therefore, the *meta-learner* is learning at a higher, meta-level, learning how to best learn across a set of tasks. We will also need to define the term of a *family of tasks*. In this instance, a family of tasks will refer to any set of tasks that can be trained to solve, each holding some fundamental relation to each other. For example, this could include new maze environments posed to a navigating robot that had learned to learn how to effectively navigate previous maze setups, or a video game player quickly learning to effectively succeed at a new game challenge it is faced with.

3 Process

The general definition of a meta-learner is as follows.[1] Assume an optimizer is to be trained for models that perform classification of objects from a set of sets of training data using standard neural networks. In other words, given the meta-training set M, M_{train}: $\{(X_{\text{i}}, c_{\text{i}})\}_{i=1}^{m}$. Each base training set within M, S_{train}: $(X_{\text{i}}, c_{\text{i}})$ can be viewed as its own set of features with inputs and labels. The goal of the meta-learner is to train each of the S_{train} base-learner models using the optimizer function trained by the meta-learner to update the weights of these models from the base weights and base gradients, in lieu of using an optimization function defined by hand. The meta-learner itself is then trained through performing its own loss calculations from the overall losses of all the base-learner models, and back-propagating through the base-learners' use of the optimizer to calculate gradients. These meta-losses and meta-gradients are then fed into an optimizer for the meta-learner to update the meta-learner weights. This optimizer, known as the *meta-optimizer*, **could** be another optimizer trained by an even higher-level meta-learner, but in common practice this would be an optimization function defined by hand.

4 Pseudocode

Basic meta-learner to train a simple *meta-learning* optimization function:[5]

```
function train(forward_model, backward_model, optimizer,
    meta_optimizer, total_train_data, meta_epochs):
  # train a meta-learner
  # meta-train loop to train optimizer function
  for each meta_epoch:
    losses = 0
    # meta-train forward pass
```

```
for inputs, labels in total_train_data:
  # do forward pass
  output = forward_model(inputs)
  # calculate loss
  loss = loss(output, labels)
  # add onto sum of losses
  losses += loss
  # backward pass; store gradients in forward model
  loss.backward()
  # use trained optimizer to update current model
  optimizer(forward_model,
            backward_model)
# take meta loss, simply, as the sum of all models' losses
meta_loss = losses
# meta-train backward pass; store gradients in forward model
meta_loss.backward()
# manually update optimizer using stored
#     gradients and meta-optimizer function
meta_optimizer.step()
```

5 Conclusion

The application of learning to learn stretches far and wide, beyond just the scope of research, into real-world implementations. Humans innately have the ability to learn to learn, and adapt previously learnt learning techniques to quickly learn to solve new tasks posed to them. At a high level, the ability to learn to learn could eventually prove to be a monumental step in advancing the scope and ability of machine learning systems.

References

[1] Ricardo Vilalta & co. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

[2] Google DeepMind. Learning to learn by gradient descent by gradient descent. *arXiv*, 2016.

[3] Chelsea Finn. Various recent meta-learning approaches. 2017.

[4] Sebastian Thrun. Learning to learn: Introduction. 1996.

[5] Thomas Wolf. From zero to research—an introduction to meta-learning. 2018.