An Introduction to Meta Learning – a path to Artificial General Intelligence

Authors: Bhavesh Laddagiri, Vivek Singhal

CellStrat AI Lab

5 Sep 2020

Abstract - Meta Learning (*aka Learning to Learn*) is about designing models which can learn and adapt to new tasks fast and efficiently without much fine-tuning. We want to design models that can learn to generalize well to new tasks, data or environments which are different from the ones in training. E.g. :

1. A game playing bot trained to play PUBG should be able to generalize to Call of Duty
2. A robot trained to walk on flat soil should be able to walk on ragged terrain
3. An image classifier trained on dogs should be able to recognize cats given a few images of what a cat looks like

Types of Meta-Learning

Meta Learning can be approached in different ways :

1. Metric-Based – Learn an efficient distance function for similarity
2. Model-Based – Learn to utilize internal/external memory for adapting (MANN)
3. Optimization-Based – Optimize the model parameters explicitly for learning quickly

.

Keywords- Meta Learning, Artificial General Intelligence, Metric-based Meta Learning, Model-based Meta Learning, Optimization-based Meta Learning

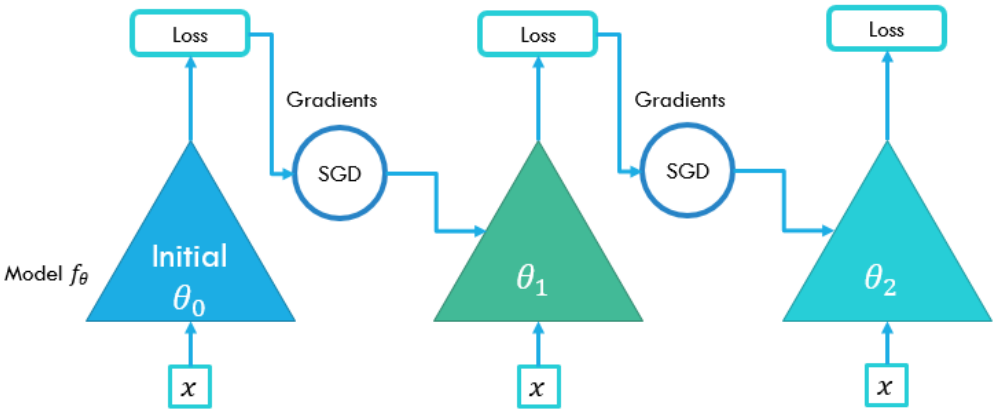
I. METRIC-BASED META LEARNING

Meta Learning helps us create generalized systems with relatively less data. It finds a lot of interesting uses :-

- Hyperparameter optimization for meta learning the optimal hyperparameters. Techniques like Genetic Algorithms can also be used to optimize the neural network hyperparameters (meta).
- Neural Architecture Search
- Few-shot learning and generalization across multiple tasks with little/no fine-tuning.
- Research in Meta-Learning is also driving the research towards AGI with advancements in Hybrid AI, Reasoning and RL.

Let’s understand a simple Meta Learning System and how it compared with Normal ML training.

Normal Training



In normal ML training, two meta factors affecting the training process are: ***Optimizer*** *and* ***Initial Parameters***.

Here we initialize the model parameters (or model weights) θ. In Forward Propagation loop, the input X is fed to the system and it moves forward; the model calculates the activations in next layers with help of model weights.

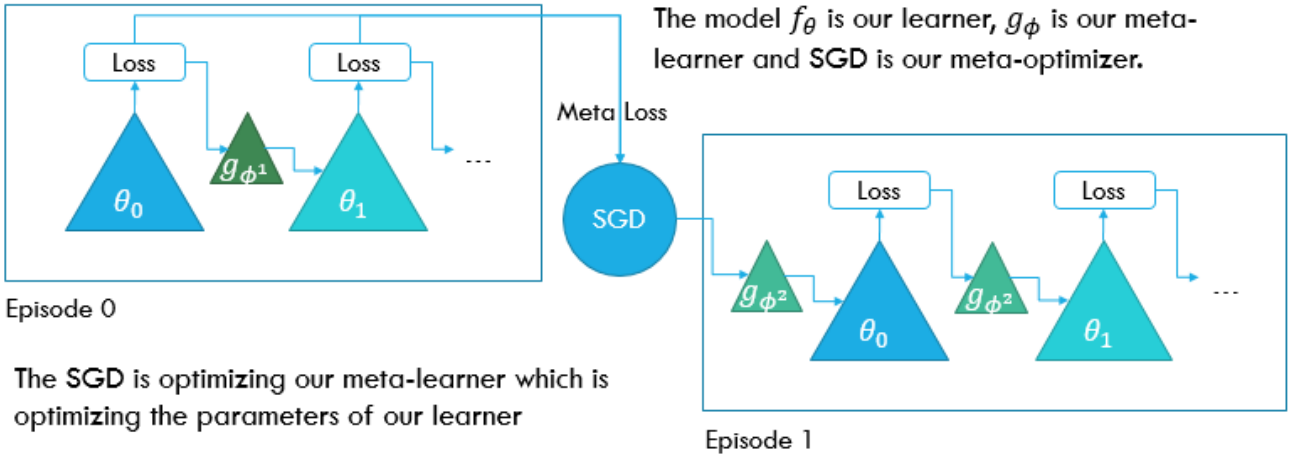
At the end of the model, we calculate the loss or cost function. This loss is back propagated and weights are adjusted with some optimization algorithm such as SGD or Stochastic Gradient Descent.

The next iteration runs with updated weights and the above cycle repeats.

Meta Learning

Meta Learning is about learning to learn, in other words learning the ideal weight initialization as well as the correct optimization logic.

One way to visualize is with this scheme :-



Here there is a g_ϕ meta-learner which in turn uses SGD meta-optimizer to learn the primary learner f_θ.

Metric-based Meta Learning

The core idea here is like the nearest neighbor algorithms (k-NN, k-means) and kernel density estimation i.e. – The predicted probabilities of a given input is equal to the weighted sum of its labels (one-hot encoded) where the weight is generated by a kernel function which measures the similarity between two samples.

$$P_\theta(y|\hat{x}, S) = \sum_{(x_i, yi) \in S} k_\theta(\hat{x}, x_i) y_i$$

Metric Learning is all about learning to measure the similarity between an input image and another image in the database (aka support set). In the next sections, we will understand the few-shot classification problem which is solved by this approach.

Few-shot Classification

Few-Shot Classification is an instance of meta-learning in a supervised context.

Let's take a simple image classification dataset D with images and its corresponding labels. Each task in D is a single data-point having one image and one label and the goal is to predict the label given a single image trained on the entire dataset.

To frame it as a meta-learning problem for generalization, we can transform this dataset into a collection of mini-datasets having only a handful images per class. We train our model on all these mini-datasets with the goal to learn to classify with little training data. Each forward pass on one mini-dataset (task) is called an *episode*.

In notational terms, this translates to the Dataset *D* being transformed into a **mini dataset *B*** each containing a Support Set *S* and Query Set *Q*.

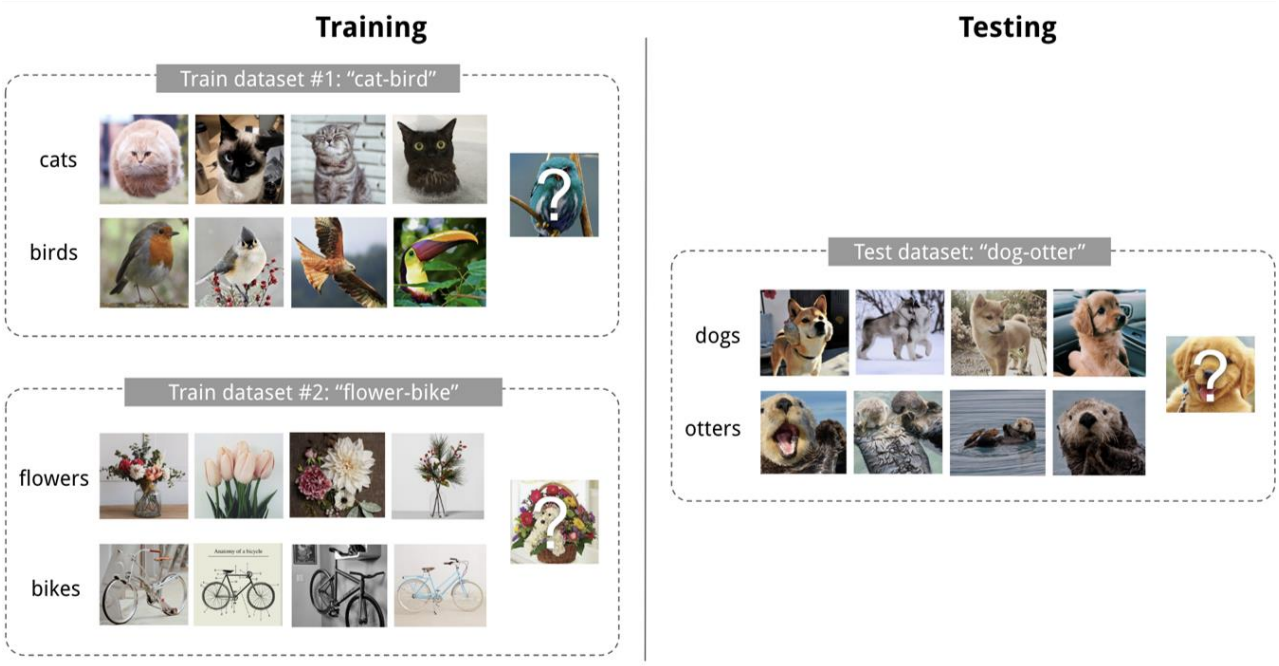
Support Set S contains the small number training examples of each class and **Query Set Q** contains the testing examples to run classification on, given the training examples in support set ***S*.**

It is modelled as a *k*-shot *n*-way problem where *k* is the number of training examples for one class and *n* is the total number of classes in the task/mini-dataset *B*. Support Set *S* contains *k* number of examples for each class in *n*.

The data transformed this way allows us to train explicitly for learning with less data.

During testing, we will be giving the model a mini-dataset *B* only, so training it in the same way as it would be tested on with mini-datasets, it is also called *"training in the same way as testing"*.

In practice, we perform similarity measurement on the support set images with the query set images making it a metric-based meta learning problem.



An example of 4-shot 2-class image classification[1]

In its simplest form, the goal is to find parameters such that

$$\theta^* = argmax\ E_{B \epsilon D}[E_{(x,y) \epsilon B}[P(y|x)]]$$

i.e. finding the optimal parameters θ^* for our model such that the expectation is maximized for all sub-tasks B in the dataset D where for each sub-task B the probability of classifying the correct label is maximized.

Other techniques of Meta Learning include ***Convolution Siamese Networks, Relation Networks, Matching Networks and Prototypical Networks***.

Key Takeaways

- Meta-Learning deals with creating models which either learn and optimize fast or models which generalize and adapt to different tasks easily.
- Few-Shot classification frames image classification as a multiple-task learning problem.
- Metric Learning is about learning to accurately measure similarity in a given support set and generalize to other datasets.

II. MODEL-BASED META LEARNING

Model-based techniques are entirely different from Metric-based techniques. They make no assumption about maximizing probability of the true class.

Instead, the focus is on making the parameter-update/adaptability on new data faster using specific model architectures which generalize.

In this section, we will look at models which are better at generalization and to some extent, reasoning too.

Memory Augmented Neural Networks (MANN)

A computer uses three fundamental mechanisms to operate – arithmetic operations, logical flow control and external memory. They have a long-term storage as well as short-term storage in the form a working memory (RAM). As humans too, we use memory to retrieve facts from past experiences and put together different facts for inference and reasoning.

But the machine learning community has largely neglected the use of external memory for solving problems. (Note: RNNs have internal memory and its different).

Making use of external memory gives extra powers to models in terms of adaptability to new tasks by just accumulating new information in its working memory.

MANNs are a class of models which learn to do just that by using attention-based mechanisms to operate on its memory and infer.

We know traditional NNs such as RNNs and LSTMs have memory concept. Unlike these, MANNs don't need to remember large amounts of data, they just need to learn to operate on an external memory and infer from it.

Usually, this is achieved by using attention-based mechanisms.

There are two types of MANNs :-

1. Memory Networks (MemNN) (not widely considered as MANN)
2. Neural Turing Machines (NTM)

Here we will review NTMs, as it has a more explicit external memory.

Neural Turing Machine (NTM)

A Neural Turing Machine is a neural network coupled with an external memory with which it interacts using attention mechanisms. It was introduced by Graves et. al. in DeepMind in 2014.

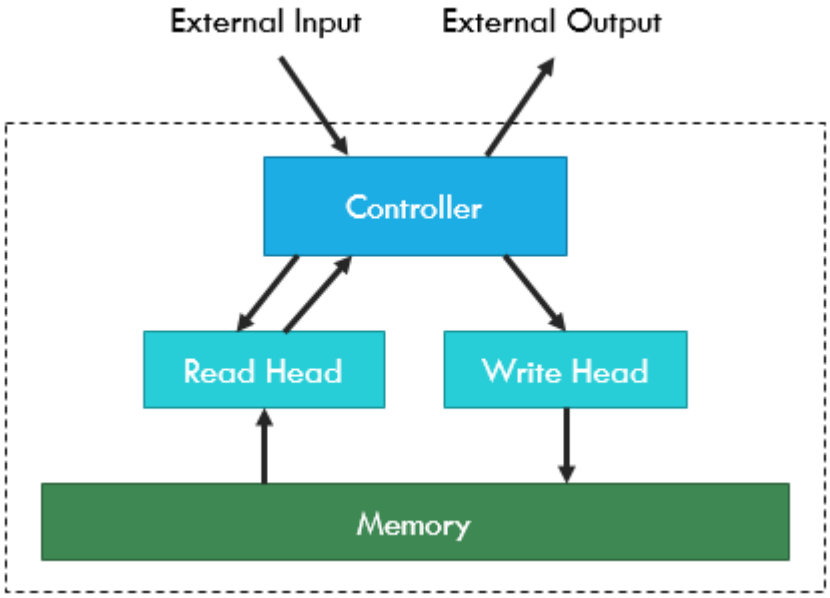
It is analogous to a Turing Machine or Von Neuman architecture except that its end-to-end differentiable, allowing it to be efficiently trained by gradient descent.

Due to the usage of an external memory, they are good at retaining long sequences and can generalize quite well.

Vanilla NTMs have been tested to learn algorithms like copy, copy-repeat, associative recall and sorting using supervised methods.

Successors like the Differentiable Neural Computer (DNC) can perform more sophisticated general tasks like route planning, answering logical questions etc.

Overview of NTM



- The NTM Architecture contains two basic components – Controller and Memory.
- The controller is a neural network which interacts with the external world with standard input and output vectors.
- But additionally, it also interacts with the memory bank using selective read and write operations. The network output neurons responsible for memory interaction are called heads (analogy of the Turing machine).
- The head interaction with memory is very sparse and it uses focused attention mechanisms.

Reading

- The memory is defined by a matrix of size N×M where, N is the number of memory locations and M is the length of the vector at each location.
- Let M_t be the contents of the memory at time t and w_t be the vector of weights over the N locations. All weights are normalized and lie between 0 and 1 and sum up to 1.
- The Read Head returns a read vector r_t of size M defined by a weighted combination of row vectors $M_t(i)$.

$$r_t = \sum_i w_t(i) M_t(i)$$

Writing

The write operation is decomposed into two parts – erase and add.

- **Erase:** Given a weight w_t emitted by the write head at time t along with an *erase vector* e_t (of length M), the memory contents $M_{(t-1)}$ from the previous time step are modified as follows,

$$\widetilde{M}_t(i) = M_{t-1}(i)[1 - w_t(i)e_t]$$

- If both weights w_t and erase vector e_t are 1 then the memory is reset to zero. If any one of them is 0 then the previous contents remains unchanged.
- **Add:** Each write head also produces an *add vector* a_t which is used to perform the changes to the Memory after the erase step.

$$M_t(i) = \widetilde{M}_t(i) + w_t(i)a_t$$

Now, we know how the read and write operations are performed by the heads, but how are these weights and parameters produced in the first place? In the next section, we will understand the addressing mechanisms which generate these parameters.

Key Takeaways

- Memory Augmented Neural Networks (MANN) use external memory coupled with attention to generalize and adapt to different tasks.
- Memory Networks use attention over the supporting information (e.g. stories) to infer on the input question.
- Multiple hops in Memory Networks can help consider multiple sequences for inference and resembles recurrent networks.
- Neural Turing Machine has controller neural network which learns to read and write to a memory matrix using selective attention to solve a task.
- NTMs tend to learn internal algorithms to solve the problem and thus generalize well.

III. OPTIMIZATION-BASED META LEARNING

Currently, all deep learning models learn by backpropagating the gradients and then performing gradient descent. But this method is not designed for learning with less data and requires a lot of iterations to arrive at the minima.

Optimization-based Meta-Learning intends to design algorithms which modify the training algorithm such that they can learn with less data in just a few training steps.

Usually, this refers to learning an initialization of parameters which can be fine-tuned with a few gradient updates. Some examples of such algorithms are –

1. LSTM Meta-Learner
2. Model-Agnostic Meta-Learner (MAML)
3. Reptile

Transfer Learning vs Optimization-based Meta Learning

For transfer learning to work, we first pretrain a model on a very large dataset. The resulting pretrained model is then used as the initial parameters for fine-tuning on a new and medium-sized dataset.

However, transfer learning is not designed specifically for learning with less data. It just happens to be good at medium-sized datasets because of the pre-learned features.

Optimization-based meta learning is aimed at finding those initial set of parameters which are generalizable to a wide range of problems, so that when we have a new problem, we only need a few gradient updates for fine-tuning to a small dataset.

These algorithms are explicitly designed and trained for finding that set of initial parameters which can be fine-tuned later in a couple of training steps.

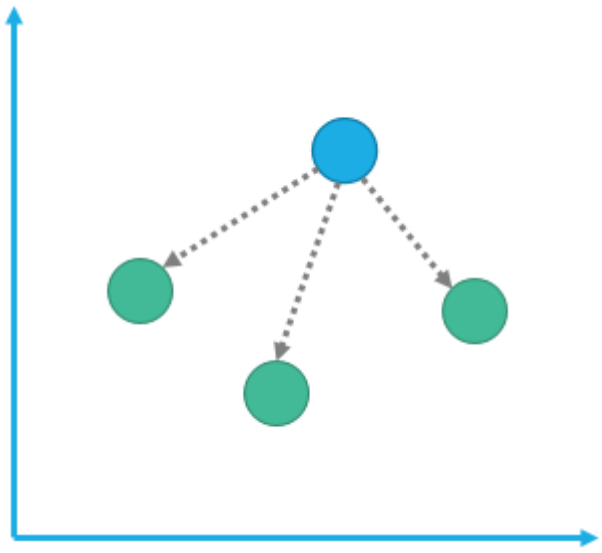
Model-agnostic Meta Learner (MAML)

It was introduced by Finn et al. (2017).

It is a model-agnostic meta-learning algorithm which explicitly learns parameters that can generalize to any new task by fine-tuning with a *single training step*.

It is model-agnostic i.e. it can work with any deep learning model which is trained with gradient descent. The authors of the original paper have tested it on few-shot classification, regression and even reinforcement learning to demonstrate its flexibility.

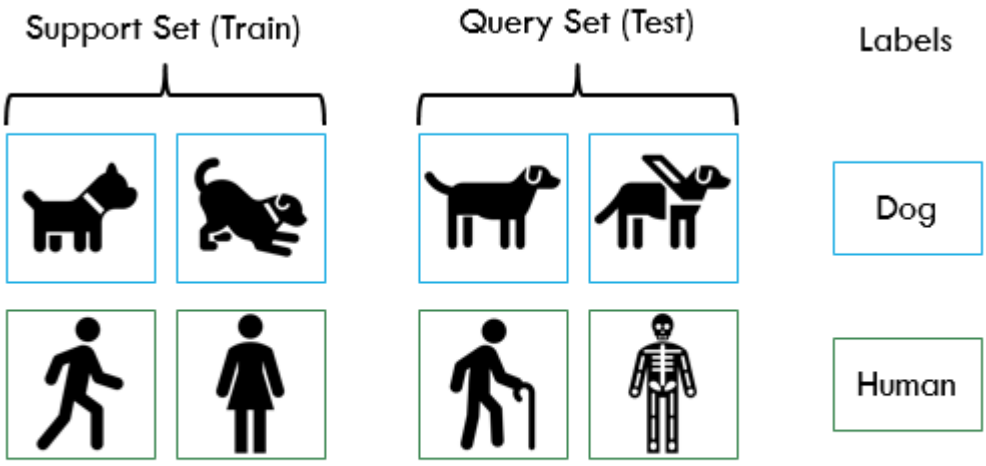
MAML Algorithm



During the meta training, we optimize to find the optimal initial parameters (blue) such that it is close to all related tasks.

Performing a few steps (usually 1) of fine-tuning on the specific task should generalize it well without overfitting.

Let’s define the problem :-



2-shot 2-way task

1. We will be applying MAML on few-shot image classification problem. The dataset is split into multiple tasks.
2. Each task is sampled as K-shot N-way classification. K images for each of the N classes which comprises of the Support Set (train).
3. The Query Set (test) also contains some K images for each of the N classes.
4. The model is a vanilla CNN based image classifier.

1. Initialize the model parameters θ
2. Start the Epoch –
    1. Sample a batch of tasks T_i
    2. For all T_i do –
        1. Sample the support set S_i
        2. Make predictions on S_i using the initial parameters θ
        3. Calculate Loss $L_{S_i}(f_\theta)$
        4. Fast weights $\theta_i' = \theta - \alpha \nabla_\theta L_{S_i}$
        5. Sample the query set Q_i for the meta update
        6. Make predictions on Q_i using the parameters θ_i'
        7. Calculate Meta Loss $L_{Q_i}(f_{\theta_i'})$
    3. Gather meta losses $L_{meta} = \Sigma L_{Q_i}(f_{\theta_i'})$
    4. Update $\theta = \theta - \beta \nabla_\theta L_{meta}$

Task Fine-tuning

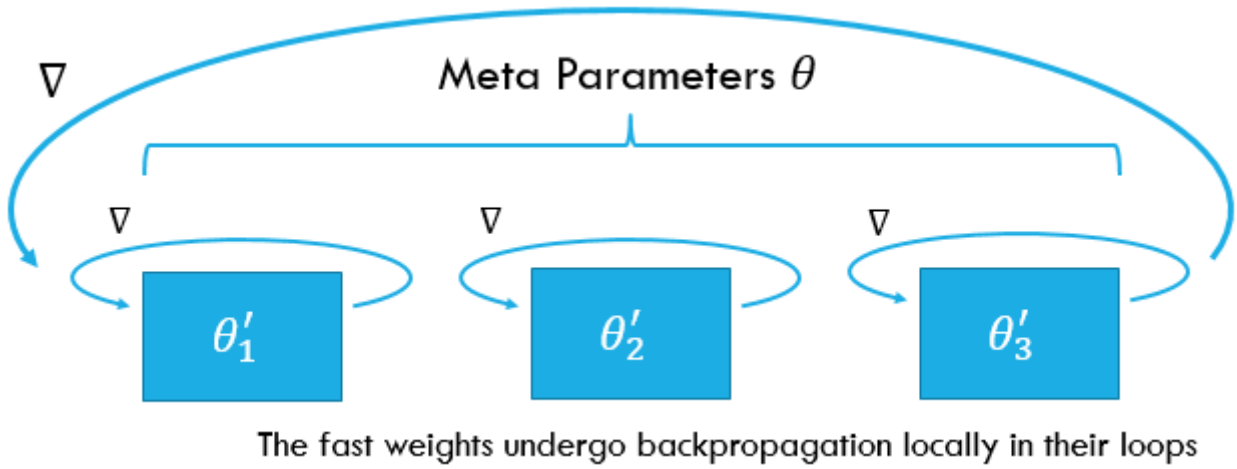
(θ_i' is called as fast weights because it is obtained with a single training step)

$$\text{Update } \theta = \theta - \beta \nabla_{\theta} L_{meta}$$

In this last step (above) the meta loss gradient is calculated with respect to the initial parameters θ making the backpropagation go through the entire computation graph including the task specific fine-tuning part where we had calculated the gradient once for support set loss L_{Si}.

This means it involves taking the gradient of the gradient making the derivative of second-order (aka Hessian vector).
Intuitively, this means moving the initial parameters θ to a place which is easy to reach for all tasks for fine-tuning in a single gradient update.

The Meta Parameter Update of θ is the backpropagation through the entire inner ***fast weights*** leading to ***second order derivatives***.



Second Order Derivatives in MAML

Applications of Optimization-based Meta Learning include :-

- Few-shot image classification
- Regression
- Reinforcement Learning
- Any deep learning model which learns uses gradient-based optimizers (Adam, SGD, etc.).

Where to use what

Metric-based Meta Learning – Use for training on datasets for few-shot classification problems. The required dataset should have enough classes to extract few-shot samples during meta-training.

Model-based Meta Learning – Memory Networks are mainly used for NLP based tasks. But it can be tuned for any sequence-based tasks. Memory Networks are shallow and compute friendly.

Neural Turing Machine can also be extended for few-shot classification problems and vanilla NTM can be used for supervised training for algorithmic tasks.

Optimization-based Meta Learning – MAML can be used for few-shot classification problems or any similar problem where data is scarce. MAML is model agnostic so it is flexible enough to fit any domain mostly. But it requires more compute power.

Usually, Meta learning is used for few-shot classification like tasks, so in general Reptile is good for that task while using lesser compute and time than MAML.

If the problem is few-shot classification, choose Reptile and if it is any other domain then choose MAML (First Order variant of MAML).

REFERENCES

[1] Meta-Learning: Learning to Learn Fast, Lilian Weng, https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html
[2] Memory Networks https://arxiv.org/abs/1410.3916
[3] End to End Memory Networks https://arxiv.org/abs/1503.08895
[4] Neural Turing Machine https://arxiv.org/abs/1410.5401
[5] https://www.niklasschmidinger.com/posts/2019-12-25-neural-turing-machines/
[6] One-Shot Learning with MANN https://arxiv.org/abs/1605.06065
[7] Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks https://arxiv.org/abs/1703.03400
[8] On First-Order Meta-Learning Algorithms (Reptile) https://arxiv.org/abs/1803.02999
[9] https://towardsdatascience.com/advances-in-few-shot-learning-reproducing-results-in-pytorch-aba70dee541d
[10] https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html#optimization-based
[11] https://towardsdatascience.com/paper-repro-deep-metalearning-using-maml-and-reptile-fd1df1cc81b0