A collection of notes

Jason Krone
Columbia University

February 9, 2018

Abstract

This is a collection of notes on machine intelligence for personal reference.

Contents

1 Model Agnostic Meta Learning by Finn et al. 2017 3

1.1 Summary: 3

1.2 Model: 3

1.3 Training: 3

1.4 Experiments: 4

1.5 My Conclusions: (concerns, follow up, etc) 4

1 Model Agnostic Meta Learning by Finn et al. 2017

1.1 Summary:

Goal: train a model on a set of learning tasks such that it can solve novel learning tasks using a small number of training samples.

Meta-learning should be general to the task and model.

MAML does not introduce more parameters or place constraints on model architecture.

MAML increases the sensitivity loss functions to new tasks i.e. small parameter changes cause large loss improvements.

1.2 Model:

Uses entire tasks as training examples.

Let a model $f_\theta : x \to a$ where x is an observation and a is an output.

A task T is defined as $T = \{L(x_1, a_1, \dots, x_H, a_H), q(x_1), q(x_{t+1} \mid x_t, a_t), H\}$ where L is a loss function, $q(x_1)$ is a distribution over initial observations $q(x_1)$, $q(x_{t+1} \mid x_t, at)$ is a transition distribution and H is the episode length.

Note: this formulation is like a reinforcement learning problem expect we assume to have the transition function?

Let $p(T)$ be a a distribution over tasks we want the model to adapt to.

1.3 Training:

1. A task $T_i \sim p(T)$
2. the model f_θ is trained with K samples using feedback from L_{T_i}
3. f is tested on new samples from T_i
4. f is improved by considering how the test error on new data from q_i changes w.r.t the parameters

In a sense the test error on sampled tasks is used as the training error for meta-learning.

Tasks used in meta-testing are held out during meta-training to remain "unbiased?"

```
**while** not done **do**
    Sample batch of tasks $T_I \sim p(T)$
    **for all** $T_i$ **do**
```

Evaluate $\Delta_\theta L_{T_i}(f_\theta)$ with respect to K examples.
Compute adapted parameters with GD: $\theta_i^{'} = \theta - \alpha \Delta_\theta L_{T_i}(f_\theta)$

The meta learning objective is as follows:

$$\min_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i^{'}}) = \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \Delta_\theta L_{T_i}(f_\theta)})$$

Meta-optimization is done as follows:

$$\theta \leftarrow \theta - \beta \Delta_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i^{'}})$$

For reinforcement learning focused problems the loss is of the form:

$L_{T_i}(f_\phi) = \mathbb{E}_{x_t, a_t \sim f_\phi, q_{T_i}} \left[\sum_{t=0}^{H} R_i(x_t, a_t) \right]$

This is used in the context of policy gradient methods, which are on policy. Since they are on policy, each additional gradient step requires sampling from the current policy $f_{\theta_i^{'}}$.

1.4 Experiments:

Comparisons are made between MAML, pre-training on all tasks, and an oracle (best possible performance).

1. Tasks: Regression from input to output of sine function where amplitude and phase of sine vary for tasks. Result: Very low MSE for MAML and very high MSE for pre-trained model.
2. Tasks: Few shot classification on Omniglot and MiniImagenet. Result: near perfect accuracy on Omniglot and best MiniImagenet accuracy by 3-5 percent.
3. Tasks: Continuous control environments for reinforcement learning where goal is to run in a particular direction or at a particular velocity. Result: outperforms pre-training by a large margin and almost reaches oracle performance.

1.5 My Conclusions: (concerns, follow up, etc)

What are important applications of MAML?

- Language models? To quickly adapt to different language pairs in translation for instance ?
- Could this be applied Bayesian models also?
- Speech recognition with different accents or languages ?

- Hierarchical reinforcement learning ? Adaption to multiple tasks and come up with a general hierarchical structure?

Concerns?

- Can this be run in real time for robotics application ?
- How much training data do you need for this method? How many tasks / samples per task?
- Is this still helpful when you have a lot of data? For example, as a pre-training method for Imagenet.

Follow up ?

- How does the model architecture effect the performance of MAML?
- Is it helpful if the model "remembers" things across tasks?
- How well does this work with recurrent / memory networks?

References