Analysis

Overseer failures in SupAmp and ReAmp

How to determine their effect on overall failure rate

Richard Möhn

11th September 2019

Contents

1 Introduction

(For a general overview and a glossary, see the home page of the Farlamp repository.)

Capability amplification can cause the failure rate of an ML system to blow up, even when the amplified agent fails rarely. To fix this, reliability amplification may be used. Alternatively, to keep the failure rate down and even decrease it below the agent's, distillation by reinforcement learning (RL) might prove sufficient. However, whether the result will be reliable enough, we don't know. (Christiano 2016b) But we do need to know in high-stakes situations (see Christiano 2016a).

If RL alone is sufficient, we can avoid the computational and design overhead of reliability amplification. Therefore, I'm planning to adapt to RL the iterated distillation and amplification scheme from Christiano et al. (2018). (See also: How to turn SupAmp into ReAmp?) Once adapted, I can investigate what overall failure rates result from various overseer failure rates and other learning parameters.

In this preliminary analysis I lay out my current understanding of overseer failure, related concepts and what I need to do. I will update it and answer open questions on the way to implementation.

2 Concepts

Before I can write about my predictions, what experiments to run and how to implement them, I will pin down some terms and concepts.

2.1 General terms

In the following, being more **capable** means being able to answer harder questions. The **overseer** is a human or some extract of human judgment. The **assistant** is a machine learning (sub-)system, which can answer the overseer's questions. In reverse, the overseer can give data to the algorithm that trains the assistant to become more capable.

2.2 Iterated amplification and distillation

The overseer alone can only provide data to train the assistant until the capabilities of the two are roughly equal. After this the training can be continued by using **Iterated distillation and amplification** (IDA). **Distillation** is the mechanism by which the assistant is trained. During **amplification** the assistant answers the overseer's questions to help it generate higher-level training data.

IDA has been explained in other places (Cotra 2018; Evans et al. 2019). But in order to write this analysis clearly I need the notion of amplification without immediate distillation. For this I don't know of any brief explanation, so I've written one:

Suppose you are an overseer H who wants to train an assistant X (symbols taken from Christiano et al. 2018) to give a solution to problem p. If you want to train X using supervised learning (SL), you ask yourself: 'What is the solution to p?' You answer: 'The solution is s.' Then (p, s) is the training datum for X. If you want to train X using reinforcement learning, X has to suggest a solution s, then you ask yourself: 'What reward should I give X for solution s to problem p?' You answer: 'The reward should be r.' Then (p, s, r) is the training datum for X.

After some time, X can solve all the problems that you can solve. But you want it to become able to solve harder problems, for which you alone can't answer the question and therefore can't provide training data. What do you do? You break the question down into sub-questions. These branch out into sub-sub-questions and so forth, forming a tree. The further you go towards the leaves, the easier the questions become. The leaf questions are so easy that you can answer them without breaking them down. Then the answers flow upwards and are combined at each node until you obtain the root answer, the missing piece of the training datum (p, s) or (p, s, r) for X. This is **iterated amplification** (IA), without distillation. (For examples of trees, see Stuhlmüller (2018b).)

However, this breaking down of questions is computationally expensive. To make it cheaper, imagine you had an assistant X^{-1} that is almost as capable as X. Then you could break down only the top-level question into sub-questions, have them answered by X^{-1}, and use these sub-answers to answer the root question. Again you've obtained a training datum for X.

This corresponds to the first iteration of iterated **distillation** and amplification: You train an agent X^0 until it is roughly as capable as you. The combination of you and the agent forms a more capable compound agent HX^0. Then HX^0 generates data to train the next agent X^1, which is more capable than X^0. This is called 'distilling' HX^0. Now you can team up with X^1 to become HX^1 and generate data to train X^2. And so on.

Now we add **failure**. It's natural that you sometimes make a mistake and return a wrong answer. What happens with IA? Assume that a wrong sub-answer anywhere in the tree leads to the root answer being wrong, and each non-leaf question leads to at least two sub-questions. Then even if your failure probability is very small, the probability of ending up with a wrong root answer approaches 1 exponentially quickly in the height of the tree. (Christiano 2016b)

In contrast, the case of I*D*A isn't clear. It depends on how the X^n are trained. If the training algorithm causes them to faithfully copy all your answers, the failure probability will compound as in IA. But if the training algorithm makes the X^n good at 'recognizing concepts', they will treat some wrong inputs as outliers and the failure probability won't compound as quickly, or it might even go down as n grows. Of course, the more mistakes you make, the harder it becomes for a learner to tell apart right and wrong. The goal of the Farlamp project is to find out how the training algorithm influences the failure rate.

2.3 Overseer determinism and non-determinism

The overseer might behave and fail deterministically or non-deterministically. Ie. it might have a fixed output for each input, and thus the same failures for a specific set of inputs. Or it might return a random value for any input, with a certain probability.

I can't rule out either way. While I tend to deduce non-determinism from the overseer model in Christiano (2016b), this deduction is countered by the section on sequential decision-making in the same article. Further, there are realistic examples of both ways: If the overseer is a learned agent, as H' in Christiano et al. (2018), it is mostly deterministic. Granted, the answers of H' change over time, because it is trained continuously on data from H. But I expect it to converge on determinism. If the overseer is a human, as in Ought's factored evaluation experiments (Stuhlmüller 2019), it is non-deterministic. Although determinism can again result if the human output is automated (see Stuhlmüller 2018a, sec. 'Caching' f.).

2.4 Height of a question

This property might become useful later in the process. The height of a question is the height of the tree the question would be decomposed into if iterated amplification was used. The following definition doesn't depend on trees and iterated amplification, but it uses the notion of primitive questions from Christiano et al. (2018, app. C).

Definition 2.1 (Height of a question)**.** By recursion:

1. Primitive questions have height 0.
2. A question has height n if $n-1$ is the maximum height of the sub-questions it is decomposed into.

Example 2.1. The decomposition rule for permutation powering from Christiano et al. (2018, table 3) is:

$$\sigma^{2k}(x) = \sigma^k(\sigma^k(x)) \tag{1}$$
$$\sigma^{2k+1}(x) = \sigma(\sigma^k(\sigma^k(x))) \tag{2}$$

- 'What is $\sigma(4)$?', has height 0, because it is a primitive question.
- 'What is $\sigma^2(4)$?', has height 1, because H decomposes it into 'What is $\sigma(4)$?' and 'What is $\sigma(\langle\text{answer to the first sub-question}\rangle)$?', which are primitive questions (height 0).
- 'What is $\sigma^3(4)$?', has height 1.
- 'What is $\sigma^4(4)$?', has height 2.
- 'What is $\sigma^8(4)$?', has height 3.

3 Predictions

Based on the previous sections I expect that if the overseer's failure probability is above a certain threshold $\hat{p}$ (call it the failure tolerance), IDA increases the overall failure probability. If the overseer's failure probability is below the threshold, IDA decreases the overall failure probability. Christiano (2016b) might have expressed this before, but I don't completely understand the relevant sections.

I expect that the threshold depends on a few parameters, such as whether H is deterministic and how X is trained. I tried to predict in which direction each parameter pushes the threshold. But so far my ideas have led me into the fog, because I know too little about the details of SL and RL.

For example, Christiano (2016b) states: 'if the overseer fails with probability 1%, then this only changes the reward function by 0.01, and an RL agent should still avoid highly undesirable actions'. Can't the same be said for SL? If only 1% of the training data for X are wrong, won't it treat them as outliers, provided enough regularization?

Note that Christiano (2016b) doesn't state anything about SL. He only predicts that distillation by imitation learning (IL) will lower reliability. Since I imagine IL to be similar to a lookup table, it makes sense that it will replicate H's failures (see sec. 2.2).

But a lookup table would not be able hold all possible input/output pairs of the tasks in Christiano et al. (2018) anyway. Instead, the learning algorithm needs to 'understand' what the tasks are about. This in turn makes it more likely to discard faulty data from the overseer.

I do think the failure tolerance $\hat{p}$ will be greater with a non-deterministic overseer than with a deterministic one. Because in the former case a randomly wrong training datum can be overridden by a correct one, but in the latter the wrong datum will be repeated – hammered in, so to say. One might argue that such repetitions are rare if the domain is large. But the domains start out fairly small in the training schedule of Christiano et al. (2018). And the start is where the foundation for the rest of the training is laid. If this foundation is faulty, correct results cannot be built.

4 Experiments

I want to measure the overall failure rate of SupAmp and ReAmp, given various overseer failure rates and maximum question heights, a non-deterministic and a deterministic overseer, and perhaps different degrees of regularization. I will decide the details later.

In the non-deterministic case, overseer failure can be modelled by injecting a random solution (SupAmp) or random evaluation (ReAmp) into the output of H' with a certain probability. One might think that not H', but H should fail, because the former is the actual overseer and the latter its imitation. But this would just degrade the learning performance of H' and not deliver randomly wrong training inputs to X as required.

The deterministic case is similar, except that a small fixed set of inputs to H' must be mapped to fixed random outputs. This can be done with a lookup table. Or by hashing the input to provide a fail/no fail value and in the 'fail' case obtaining a fixed random output by hashing again.

References

Christiano, Paul (2016a). *Learning with catastrophes.* URL: https://www.alignmentforum.org/posts/qALeGJ9nPcs9eC9Af/learning-with-catastrophes (visited on 2019-09-04).

– (2016b). *Reliability amplification.* URL: https://www.alignmentforum.org/posts/6fMvGoyy3kgnonRNM/reliability-amplification (visited on 2019-09-02).

Christiano, Paul, Buck Shlegeris, and Dario Amodei (2018). 'Supervising strong learners by amplifying weak experts'. In: arXiv: 1810.08575 [cs.LG].

Cotra, Ajeya (2018). *Iterated Distillation and Amplification.* URL: https://www.alignmentforum.org/posts/HqLxuZ4LhaFhmAHWk/iterated-distillation-and-amplification-1 (visited on 2019-09-06).

Evans, Owain, William Saunders, and Andreas Stuhlmüller (2019). *Machine Learning Projects for Iterated Distillation and Amplification.* URL: https://owainevans.github.io/pdfs/evans_ida_projects.pdf (visited on 2019-09-10).

Stuhlmüller, Andreas (2018a). *A taxonomy of approaches to capability amplification.* Ought. URL: https://ought.org/research/factored-cognition/taxonomy (visited on 2019-09-05).

– (2018b). *Factored Cognition.* URL: https://www.alignmentforum.org/posts/DFkGStzvj3jgXibFG/factored-cognition (visited on 2019-09-06).

– (2019). *Delegating open-ended cognitive work.* URL: https://ought.org/presentations/delegating-cognitive-work-2019-06.