Research proposal

Introduction

Transfer learning for reinforcement learning.

What is reinforcement learning (RL)?

An easy setting to understand RL is in the Markov decision process (MDP) setting.

A MDP is defined as a tuple, $\{\mathcal{S}, \mathcal{A}, P(s_{t+1} \mid s_t, a_t), R(s_t, a_t, s_{t+1})\}$. Where $\mathcal{S}$ is the set of possible states (*for example arrangements of chess pieces*), $\mathcal{A}$ is the set of actions (*the different possible moves, left right, diagonal, weird L-shaped thing, …*), $P(s_{t+1} \mid s_t, a_t)$ is the transition function which describes how the environment acts inresponse to the past and to your actions (*in this case, your opponent's moves, and the results of your actions*), and finally, $R(s_t, a_t, s_{t+1})$ is the reward function, (*whether you won (+1) or lost (-1) the game*), which you are trying to maximise.

This setting is easily generalised to other, more interesting, applications. For example, if we weaken the requirement on that the learners observation x_t is fully describes the state s_t, then we could apply this framework to partially-observable decision processes, such as StarCraft II or self-driving cars.

(originally inspired by research in physchology? Thus 'good' actions can be reinforced by receiving positive rewards, while 'bad' actions can be punished.

It should be noted that the problems that make RL hard are; "trial-and-error search and delayed rewards" [Sutton and Barto]. Unlike supervised learning, which gives the learner feedback (*I think that digit is a 5 -> no, its a 6*), in RL the learner only receives evaluations (*I think that digit is a 5 -> wrong*).

What is transfer learning (TL)?

An easy setting to understand TL is in the multi-task setting (but there are others, one-shot and zero-shot learning, continual learning and even distribution shift could be viewed as transfer learning between the past and present)

Imagine we have two classification tasks, A, B, which each consist of pairs of observations and targets task $= \{(x_i, t_i) : i \in [1 : N]\}$. Now, a learner, f, trained on task A and achieves a loss, $\mathcal{L}$, when evaluated on task A. We denote this as: $\mathcal{L}^A(f_A)$ (subscript denotes training task, superscript denotes evaluation task).

We say transfer has occured when $L^A(f_A) > L^A(f_{B \to A})$ (goal is to minimse loss). That is to say, that training on B, and then training on A imporves

performance on A. This is also known as pretraining. Similarly, we could transfer in the opposite direction, $L^A(f_A) > L^A(f_{A\to B})$. Training on A and then training on B improves performance on the original task, A. (note this is actually quite hard ref!?)

An example of transfer is !!!

But there is more to it than that? Forward vs backward transfer. Many applications, … Needs to be safe! What about efficient transfer? Increasing the speed or learning (aka meta learning).

Captures the idea of learning something more abstract about the similarities between two distinct tasks. Closely related to the notion of generalisation.

Why do we care?

Applications. Existing research. Blah Blah.

Transfer as decomposition

One way to achieve transfer learning is to: decompose complex observations into a set of modules/atomic-factors. Or in other words, to disentangle independent factors. (need refs!!!)

For example;

- take a symmetry group, can be rewritten as the composition of a single atom with various transformations. Thus we have decomposed the symmetry into is parts.
- Aka finding the basis.
- Decomposition of objects and relations?

The belief that complexity can be decomposed, that it can be built from smaller/few parts, is at the heart of unsupervised learning. For example, it is comon to assume that there are a set of latent variables that combine (non)linearly to generate the observed complexity. (and also science - reductionism!? gush about science! and automated science?)

Although, should be noted that, even if we have a great decompostion of a complex phenomena, there is a lot of meta information required to actually use this decomposition. Which modules do what? How are they related? When should I apply a given module?

Note, there are other approachess to transfer(?). Learning a metric? Distillation? EWC? MDL regularisation? ??? (but maybe underneath they are all related?!?)

Two ways to enforce a decomposition; structural constraints or a regulariser (?). (other ways?)

We can build a decomposition into the strucutre of our model, for example, we can define a transition function that optimises next step prediction error, and a policy that is optimised via its correlation with reward. Another example could be ensembles, where to get a decomposition of the learners we can feed them different inputs, forcing them to do different things. (this requires domain knowledge!?)

Or we can regularise a general function approximator to build disentangled representations. For this we needa a differentiable measure of disentanglement.

Decompositions in RL

Let's explore some decompositions in RL.

Model-based RL can be viewed as the decomposition of a reinforcement learner into a transition function and a policy. This facilitates transfer as, the learner can be given a new task, requrining a new policy, (while remining in the same environment) and simply reuse its model. Notably model-free learners cannot do this. (refs!) If we could get model-based RL to work, it would take use a long way to solving many problems in XXX?. But, in the real world, our ability to learn and transfer models is still an unsolved problem. Which will require a large amount of transfer itself.

There are other examples of existing work attempting to use a decomposition to facilitate transfer, for example;

- Feudal networks decompose the policy into a manager and a worker who work a different time scales.
- Learning to learn by GD by GD decompose learning into, learning a teacher and a student
- Modular meta-learning

Measure How do we know if two phenomena should be decomposed? What measures are there that tell use the relationship between two variables? Define decomposition: ? Define disentangled representation: ? hmm. Hypothesis, we are missing a coherent definition that captures our intuitive notion of decomposition/disentanglement.

As well as **Modular** ML , ,

Ensembles … Outrageously large networks

Heirarchies are one of the most important ideas being used in machine learning is that of a heirarchy. The idea of composing simple conecpts into more complex ones, … For example, a heirarchy of value functions (long term planning), Heirarchy of actions .

Setting

The current state-of-the-art in TRL (might be) DARLA

Where TRL can be applied is conditional on where RL can be applied. Thus if we want to scale TRL to the 'real world', we first need to extend RL to work in the real world. Some of the key differences between the places RL has been successful (for example; Go, Atari ALE) are;

- continuous actions (especially important for robotics),
- resource constraints (online learning),
- partial information (where model-based RL becomes necessary – ?),
- long-term

As well as, requiring massive amounts of data, and access to simulators. (but transfer learning hopes to solve these problems) What about safety?

One of the reasons, in my opinion, that the machine learning field has progressed quickly is its use of shared benchmarks on currently out-of-reach problems. ImageNet is a great example of this for the computer vision community, and the Atari ALE is currently the canonical benchmark for the RL community.

There is a need for cleverly designed benchmarks that dont take thousands of CPU/GPU hours to evaluate on. TRL is a game for players with resources. By definition, it requires the testing of algorithms on many different tasks.

What is really needed for the TRL community are some just-out-of-reach benchmarks to measure progress? I am imagining a set of as simple as possible settings where …, similar to ai-safety-worlds.

The field lacks a clear benchmark for transfer/decomposition. ??? Find and design toy problems! Show that existing approaches cannot do X.

- learn periodic table from chemistry data (?)
- ?

Proposed research

Goal of research is ??? understanding! How to get understanding? A necessary part is building it and showing it work.

Specific questions to explore

(these may be ill-posed, trivial, or solved, but hopefully I will find out soon…)

Fundamental RL

- What can you learn from an interactive environment (you can take actions and observe their results) that you cannot learn from a (possibly comprehensive) static dataset? Similarly, what can you learn when you have access to an ϵ-accurate model, that you cannot learn from an interactive environment?
- Why does distributional RL provide such a large advantage?
- Efficient search. Relationship between tree search and gradient descent? Local and global optimisation.
- Temporal credit assignment.

Decompositions

- Explore the heirarchical composition of linearly solvable markov decision processes Saxe et al. 2016
- Does a temporal decomposition (moving averages at different scales) of rewards produce a generalisation of meta-RL learning?
- Model the transition function as a mixture of densities, $s_{t+1} = \text{argmax}_{s_{t+1}} \prod_i p_i(s_{t+1} \mid s_t)$.

Unsupervised learning

- How can we learn decomposed representations? ICA, Lateral inhibition, residuals?
- Inverse energy learning. Similar to inverse reinforcement learning what if we assume that the observations we make are the results of an energy being minimised, $\Delta x = -\eta \frac{\partial E}{\partial x}$. Thus we could try to learn E.
- MLD? symmetry strucutre? What are the 'right' priors? How can we optimise them?

Model-based learning (with partial information)

- Build a differentiable neural computer with locally structured memory (start with 1d and then generalise to higher dimensions).
- Is the ability to localise oneself necessary to efficiently solve partial information decision problems?
- The exploration policy may influence the dynamics observed, thus we need to correct for the off-policy actions. (The model must learn an approximation of the policy being followed.)

Planning with a learned model (with continuous actions)

- If a model is being learned online how can we efficiently update value estimates computed using the old model?
- How can you backpropagate gradients through the argmax functions required for planning?
- Exploitable models. Reverse, accuracy, time step, …
- If I am using an imperfect learned model to generate plans, how can I ensure that I do not plan for 'fantastic' outcomes (aka they are fantasies).

Generalisation/transfer!?!?

- online adaptation to sensor and actuator failure.
- Is decomposition necessary for transfer?
- Distribution shift!?

While TRL is the goal. Progress in TRL is dependent on

I will not attempt to explore all of these. Rather I will take a some what random walk through them, attempting to make progress where possible. There are a few dependencies between them, for example, it is hard to plan without a model, thus planning is dependent on having models in the first place.

The goal will always be to; show that the proposed problem is actually a problem, design the minimal viable experiment to test new ideas, test new ideas…

Proposed deliverables

- Tutorial(s) on 'core' RL
- A model-based learner that is performant in the Atari ALE (or similar),
- Essay on the future of model-based RL,
- Definition and construction of a new benchmark for T(R)L,
- Implementation/Reproduction of a TRL paper

Hopefully a few papers, but that is conditional on making discoveries.