Research proposal

Efficiently learning models (for planning) via decomposition and transfer

Alexander Telfar

2018-11-1

Introduction

Popular opinion within the machine learning community is that transfer reinforcement learning (TRL) is the most likely path to achieving general artificial intelligence (Legg and Hutter 2007) (Higgins et al. 2017). Is this really true? And if it is, how can we do it? Let's explore the claim and some approaches.

What is reinforcement learning (RL)?

RL is often associated with the carrot-and-stick metaphor. Good behaviour is rewarded with carrots, while bad behaviour is punished with sticks (which surely taste bad). There are two main features of RL that make it hard: "trial-and-error search and delayed rewards" (Sutton and Barto 1998). Unlike supervised learning, which gives the learner feedback (*I think that digit is a 5 -> no, it's a 6*), in RL the learner only receives evaluations (*I think that digit is a 5 -> wrong*). Ontop of terse teachers, many actions may be taken before any evaluation is received, thus requiring credit to be assigned to actions, often leaving the learner wondering: "what did I do to deserve this?" (see pigeon superstition for an amusing example of credit assignment gone wrong (Skinner 1948)). An easy way to understand the formal RL setting is as a Markov decision process (MDP).

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 $s \in \mathcal{S}$ is the set of possible states (*for example arrangements of chess pieces*), $a \in \mathcal{A}$ 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 in response to the past (s_t) and to your actions (a_t) (*in this case, your opponent's moves, taking one of your pieces, 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*) and $R = \sum_{t=0}^{T} r(s_t, a_t, s_{t+1})$ is the cumulative reward, or return. The player's goal, is to learn a policy π, (which chooses actions, $a_t = \pi(s_t)$) that yields the largest return (max R).

This setting is easily generalised to other, more interesting, cases. For example, if we weaken the requirement on that the learners observation x_t 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, where we see only a small portion of the truly complex environment.

What is transfer learning (TL)?

TL is closely related to the notion of generalisation, when we have some existing knowledge that we repurpose (or generalise) for another use.

> A hallmark of human cognition is learning from just a few examples. For instance, a person only needs to see one Segway to acquire the concept and be able to discriminate future Segways from other vehicles like scooters and unicycles. Similarly, children can acquire a new word from one encounter (Carey & Bartlett, 1978). How is one shot learning possible? (Lake et al. 2011)

A familiar example to some is learning to eat with chopsticks (after already knowing how to use cutlery). We do not need to relearn to use our arms, we remember where our mouths are, and how to chew (we transfer that knowledge). We also remember that the goal of eating is to put food, not too much, into our mouths (we transfer that knowledge). We also transfer the knowledge of "how one might pick something up". But in this case, we need to learn some new grips and fine motor skills that will allow us to pick up the tempting food in front of us.

An easy setting to understand TL is in the multi-task setting (but there are others, few-shot learning (Lake et al. 2011), continual learning (Thrun and Mitchell 1995) and even distribution shift could be viewed as transfer learning between the past and present). Imagine we have two classification tasks, A, B, each of which 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, L, when evaluated on task A. We denote this as: $L^A(f_A)$ (subscript denotes training task, superscript denotes evaluation task).

We say transfer has occurred when $L^A(f_A) > L^A(f_{B\to A})$ (the goal is to minimse loss). That is to say, that training on B, and then training on A imporves performance on A (aka pre-training (Erhan et al. 2010)). 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, for example see Elastic Wights Consolidation in (Kirkpatrick et al. 2016)).

Why do we care?

Whether we imagine a production robot that must package an oddly shaped present, a self-driving car that must react to a clown suit running across the road, an algorithm managing an electrical grid when a storm hits, … transfer allows these agents to generalise from past experience and reliably achieve our goals. The robot "reuses" wrapping strategies from many differently shaped objects to construct a new policy. The self-driving car "guesses" that the clown suit might have a person in it since it has "similar" proportions. The grad manager makes a lucky guess – informed by pervious storms.

Without the transfer of knowledge between domains, the robots, algorithms and automaton we design would each have to start from nothing, no prior knowledge, or from human provided domain knowledge. Learning with no priors is expensive (it requires a lot of trail and error) and writing down priors that work is expensive (humans are not cheap, and often we are not capable of writing down the "right" policy).

Thus, to *efficiently* scale artificial intelligence to the real world and all of its complexity (high dimensionality, low entropy, non-linearity and non-locality), it will be necessary for knowledge to be efficiently transferred between domains.

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 (see (Gretton et al. 2007)). For example we could take a symmetry group, and rewrite it as the composition of a single element with only a few transformations. Thus we have decomposed the group into is parts.

Note: There are other approaches to transfer learning, for example; learning a metric (Vinyals et al. 2016), distillation (Hinton, Vinyals, and Dean 2015), the freezing of parameters (Kirkpatrick et al. 2016). But maybe underneath their superficial differences they are all doing the same thing, decomposing the problem into distinct parts?

The belief that complexity can be decomposed, that observations are built from smaller/few parts, is at the heart of unsupervised learning, and of science. For example, it is common to assume that there are a set of latent variables that combine (non)linearly to generate observations. And within science, reductionism and mediation analysis are key tools for producing understanding. They allow us to break down (emergent) complexity into simple parts, which we can study and understand (Wu and Tegmark 2018).

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

How to decompose?

As far as I know, there are two ways to build priors into a learner; explicitly or implicitly. Explicit priors normally mean structural constraints and regularisers. Implicit priors can mean the (unintended) priors lurking within an optimisation algorithm (for example stochastic gradient descent's bias towards flat minima (Poggio et al. 2017)).

We could build a decomposition into the structure of our model, for example, by specifying that different loss functions apply to subsets of parameters (like how the generator and discriminator in a GAN learn from different losses). Another example could be an ensemble: we feed the participants in the ensemble certain (different) features, forcing them to do different things. Often, these structural choices come from domain knowledge (beliefs we have, as people), about how the task should be done.

Alternatively, we can regularise our model to decompose its inputs. For this we need a differentiable measure of disentanglement and a flexible representation. This approach is appealing as it should require less domain knowledge.

Decompositions in RL

As noted above, a fundamental decomposition used in RL is the model-based RL framework. Where a reinforcement learner is decomposed into a transition function (a model of the environment) and a policy (how to act in the environment). This facilitates transfer as, the learner can be given a new task, requiring a new policy, (while remaining in the same environment) and simply reuse its model. For example, asking a learned-chess-master to now lose a game of chess on purpose (notably model-free learners like DQN (Mnih et al. 2015) cannot do this).

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

- Feudal networks (Vezhnevets et al. 2017) decompose the learner's policy into a manager and a worker who work a different temporal scales.
- Learning to learn by gradient descent by gradient descent (Andrychowicz et al. 2016) decomposes learning into, learning the teacher (or optimiser) and the student.
- Path net (Fernando et al. 2017) and Modular meta-learning (Alet, Lozano-Pérez, and Kaelbling 2018) both construct a single function from many modules and must learn: the modules and when to use them. They use simulated annealing and evolutionary strategies respectively to train the choice of modules.

It is possible to structure decompositions in many ways; hierarchically, as an ensemble, as a graph structure, … As of yet, there is no general formula that tells us when a decomposition of a learning problem is going to be optimal or even "good". And. They can be expensive to train, for example, (Andrychowicz et al. 2016) requires an entire neural network to be trained for each training step of the teacher.

Proposed research

Ideally, this Masters would provide an understanding of generalisation and transfer. I think that an understanding of, or theory of, generalisation and transfer would answer questions such as;

- what, when and how can knowledge be transferred from A to B?
- is decomposition necessary for transfer? Is it sufficient?
- how (computationally) hard is it to find symmetries between two domains?
- what problems can a "decomposer" solve that another cannot?

- which representation of knowledge is optimal for a given problem domain?
- how can "algebras" for composing modular systems be learnt?
- why do neural networks naturally transfer knowledge between tasks?

But I am unsure how to approach these questions and which tools are the best for thinking about them. While mulling these over, I would like to explore decompositions and model-based learning. While exploring, my hope is to see patterns and find interesting problems for understanding transfer more generally.

Benchmarks

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 (Deng et al. 2009) is a great example of this for the computer vision community, and the Atari ALE (Machado et al. 2017) is currently the canonical benchmark for the RL community.

By definition, TRL requires the testing of learners on many different tasks, which can take thousands of CPU/GPU hours to evaluate… (this is worse than other types of machince learning due to the use of hungry function approximators, neural networks, and the sparsity of the supervision, rewards). I think there is a need for benchmarks that do not require thousands of dollars to bench. Currently TRL is a game for players with resources, which means that only Google and Google-Deepmind get to play.

I am imagining a set of as-simple-as-possible settings where we can explore the limits of transfer and decomposition (similar to ai-safety-worlds for the AI safety community).

Setting

Where TRL can be applied is dependent on where RL can be applied. Thus if we want to scale TRL to the "real world", we first need to extend RL to the real world. Some of the key differences between the places RL has been successful so far (for example; Go, Atari ALE (Machado et al. 2017)) and the real world are;

- continuous action spaces (especially important for robotics),
- resource constraints (memory - online learning, speed - real-time machine learning, …),
- partial information (where model-based RL becomes necessary),
- long-term dependencies

Finally, the real world is just a lot more complex than these games we test our algorithms on. If we want to scale to the real world, we need to develop algorithms that are more data efficient. Transfer learning is a solution.

Specific questions to explore

The directions and questions set out below are my attempt to combine these motivations into some actionable research directions and questions.

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

Decompositions

1. Can we formalise what we mean by "decompose"? Can we differentiate it (so we can use it within the deep learning framework)? What is its relationship to independence criterion and independent component analysis?
2. Meta-RL (Wang et al. 2017) trains a learner on the aggregated return over many episodes (a larger time scale than typical). If we construct a temporal decomposition (moving averages at different time-scales) of rewards and approximate them with a set of value functions, does this naturally produce a rich set of options (/hierarchical RL)?
3. Imagine you are given two models f, g that (say) predict pedestrian and traffic behaviour respectively. How can you safely and/or sensibly combine their predictions? Can an advantage be gained if the models are provided as densities (rather than step functions)?
4. When learning, we want to know if existing knowledge is useful for a new task. How can we assign credit to a given "module" of knowledge? Is causal (Pearl 2018) credit assignment necessary or sufficient?

Model-based learning (with partial information)

5. Build a differentiable neural computer (Graves et al. 2016) 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? Under which conditions does the learned index to a locally structured memory approximate the position of the agent in its environment.
6. When attempting to learn a model, the agent uses an exploration policy. This policy may influence the dynamics observed, thus we need to use off-policy methods to correct for the effects of exploration actions. (The model must somehow disentangle the agents policy, and its effects, from the dynamics of the system)
7. Inverse energy learning. Inspired by inverse reinforcement learning (Ng and Russell 2000), what if we assume that the observations we make are the results of some optimal action, in this case, of an energy function being minimised.
8. While learning a model $s_{t+1} = \tau(s_t, a_t)$ is useful. It is more useful to know how to get around using that model, the reachability of various states. For example, I want to get to s^k, how can I

do that considering I am in another state, s^i?

Planning with a learned model (with continuous actions)

(Unfortunately I am not going to make it to these problems in this Masters. But I think it is important to remember the ultimate goals of TRL. We do not just care about learning models, additionally, they need to useable for efficient planning.)

- If a model is being learned online how can we efficiently update past value estimates computed using the old model?
- How can you efficiently backpropagate gradients through the argmax functions required for planning?
- 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 (note, this is closely related to reward hacking).
- Construct a planner that can control the models computational complexity by asking it to provide more approximate solutions (via accuracy masks, for the size of time-steps, the locality of interactions, …).

Timeline

I have allocated time for 8 "sprints" (the bullet points above; decompositions and model-based learning), each of 2 weeks. The goal of each sprint will be to;

- Motivate the idea as a solution to an existing problem,
- Demonstrate that the "existing" problem really exists,
- Generate alternative solutions and a suitable baseline,
- Design the minimal viable experiment to falsify the proposed solution,
- Implement the experiment if feasible.

Future work will depend on the results of the sprints.

Proposed deliverables

- Tutorial(s) on "core" RL,
- A learned model that benefits from transfer between environments (on Atari ALE or similar),
- An essay on the future of model-based RL,
- The definition and construction of a new benchmark for TL,
- A thesis documenting my work.

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

References

Alet, Ferran, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2018. "Modular Meta-Learning." *CoRR* abs/1806.10166. https://arxiv.org/abs/1806.10166.

Andrychowicz, Marcin, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. "Learning to Learn by Gradient Descent by Gradient Descent." In *NIPS*. https://arxiv.org/abs/1606.04474.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. "ImageNet: A Large-Scale Hierarchical Image Database." In *CVPR09*. http://www.image-net.org/.

Erhan, Dumitru, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. "Why Does Unsupervised Pre-Training Help Deep Learning?" In *Journal of Machine Learning Research*. http://www.jmlr.org/papers/v11/erhan10a.html.

Fernando, Chrisantha, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. "PathNet: Evolution Channels Gradient Descent in Super Neural Networks." *CoRR* abs/1701.08734. http://arxiv.org/abs/1701.08734.

Graves, Alex, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, et al. 2016. "Hybrid Computing Using a Neural Network with Dynamic External Memory." *Nature* 538: 471–76. https://www.nature.com/articles/nature20101.

Gretton, Arthur, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, and Alexander J. Smola. 2007. "A Kernel Statistical Test of Independence." In *NIPS*.

Higgins, Irina, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. 2017. "DARLA: Improving Zero-Shot Transfer in Reinforcement Learning." In *ICML*. https://deepmind.com/research/publications/darla-improving-zero-shot-transfer-reinforcement-learning/.

Hinton, Geoffrey E., Oriol Vinyals, and Jeffrey Dean. 2015. "Distilling the Knowledge in a Neural Network." *CoRR* abs/1503.02531. https://arxiv.org/abs/1503.02531.

Kirkpatrick, James, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, et al. 2016. "Overcoming Catastrophic Forgetting in Neural Networks." *CoRR* abs/1612.00796. http://arxiv.org/abs/1612.00796.

Lake, Brenden M., Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. "One Shot Learning of Simple Visual Concepts." In *CogSci*. https://cims.nyu.edu/~brenden/LakeEtAl2011CogSci.pdf.

Legg, Shane, and Marcus Hutter. 2007. "Universal Intelligence: A Definition of Machine Intelligence."

CoRR abs/0712.3329. http://arxiv.org/abs/0712.3329.

Machado, Marlos C., Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. 2017. "Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents." *CoRR* abs/1709.06009.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. "Human-Level Control Through Deep Reinforcement Learning." *Nature* 518: 529–33. https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf.

Ng, Andrew Y., and Stuart J. Russell. 2000. "Algorithms for Inverse Reinforcement Learning." In *ICML*. https://ai.stanford.edu/~ang/papers/icml00-irl.pdf.

Pearl, Judea. 2018. "Theoretical Impediments to Machine Learning with Seven Sparks from the Causal Revolution." *CoRR* abs/1801.04016. http://arxiv.org/abs/1801.04016.

Poggio, Tomaso A., Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. 2017. "Theory of Deep Learning III: Explaining the Non-Overfitting Puzzle." *CoRR* abs/1801.00173. https://arxiv.org/abs/1801.00173.

Skinner, By B. F. 1948. "Superstition in the Pigeon." *Journal of Experimental Psychology* 38 2: 168–72. https://psychclassics.yorku.ca/Skinner/Pigeon/.

Sutton, Richard S., and Andrew G. Barto. 1998. "Reinforcement Learning: An Introduction." *IEEE Transactions on Neural Networks* 16: 285–86. https://drive.google.com/file/d/1opPSz5AZ_kVa1uWOdOiveNiBFiEOHjkG/view.

Thrun, Sebastian, and Tom M. Mitchell. 1995. "Lifelong Robot Learning." *Robotics and Autonomous Systems* 15: 25–46. https://www.sciencedirect.com/science/article/pii/092188909500004Y?via%3Dihub.

Vezhnevets, Alexander Sasha, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. "FeUdal Networks for Hierarchical Reinforcement Learning." In *ICML*. https://arxiv.org/abs/1703.01161.

Vinyals, Oriol, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. "Matching Networks for One Shot Learning." In *NIPS*. https://deepmind.com/research/publications/one-shot-learning-memory-augmented-neural-networks/.

Wang, Jane X., Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. 2017. "Learning to Reinforcement Learn." *CoRR* abs/1611.05763. https://arxiv.org/abs/1611.05763.

Wu, Tailin, and Max Tegmark. 2018. "Toward an AI Physicist for Unsupervised Learning." https://arxiv.org/abs/1810.10525.