Università della Svizzera italiana

Faculty of Informatics

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Technical Report - Efficient Computational Algorithms, Fall 2020

Reproducibility challenge ATML SA 2021
Generalization in Reinforcement Learning

Erick Franciskus[†], Glejdis[‡], Constantin[‡], Jitin[‡]

[†] Faculty of Informatics, Università della Svizzera italiana, Switzerland
[‡] Faculty of Engineering, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Abstract

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus.

Report Info

Published
November 2021

Course
Efficient Computational Algorithms
Fall 2020

Institutions
Faculty of Informatics
Università della Svizzera italiana
Lugano, Switzerland
&
Faculty of Engineering
Friedrich-Alexander Universität
Erlangen-Nürnberg, Germany

1 Introduction

Reinforcement Learning is used in sequential decision making for training agents in complex tasks. The agent interacts with the environment which, based on the agent's action, provides the agent with some reward (positive or negative). The agent then uses this feedback to update its behaviour with the goal of being optimal. Although current RL agents have demonstrated great potential in a variety of activities, they face difficulties in transferring these agent's capabilities to new unseen tasks. This happens even when the tasks are semantically equivalent. In other words, existing reinforcement learning agents frequently learn policies that do not generalize well to environments different than those environments these agents are trained on. For example, the paper considered a jumping task, where we have an agent that needs to jump over an obstacle. The agent learns from image observations. In order for the agent to not collide with the obstacle, the agent needs to precisely time the jump at a particular distance from the obstacle. The various tasks consist of shifting the floor height or changing the obstacle position or both. If we train deep RL agents on some of these tasks where we vary the obstacle positions, they perform poorly at previously unseen locations (i.e. struggle to jump over the obstacle without hitting it). The challenge tackled in the paper is to generalize to unseen positions of the obstacle and floor heights in the test tasks, while using only a finite number of training tasks or environments sampled from a distribution of tasks.
A few proposed solutions to poor generalization, adapted from supervised learning, include regularization (such as l2-regularization, dropout or noise injection), Domain Randomization and Data Augmentation (such as RandConv, RAD, DrQ), which is being used more recently. The majority of these ideas centre around improving the learning process in order to fully harness decision-making process, and they rarely

use properties of the sequential aspect such as similarity in actions across temporal observations. Instead, this paper takes advantage of the fact that when an agent operates in similar tasks, the agent shows at least short sequences of behaviours that are similar across these tasks. In their approach, the authors train the agents to learn a representation where the states are close when the optimal behaviour of the agent in the current states and future states are similar. They use the notion of behavioural similarity as it has the property of good generalization to observations across different tasks. This behavioural similarity between states across various tasks is quantified using the Policy Similarity Metric (PSM), which is a state-similarity metric based on bisimulation. Moreover, to enhance generalization, the agent is trained to learn state embeddings. This is done using Contrastive Similarity Metric (CSM).

2 Theory and Concepts

2.1 The π-bisimulation and Policy Similarity Metric

2.2 Policy Similarity Embeddings - PSEs

Aiming for generalization of learned policies to related environments, the authors represent states in an embedding space, where closeness of states correlates with similar optimal policies. The embedding is built by self-supervised contrastive learning. Self-supervised learning refers to learning techniques that work without explicit annotations of the data. Instead, feedback signals are generated from the data itself. In this case this feedback comes from the policy similarity metric. [FB AI link] The contrastive instance of self-supervised learning works with positive pairs that are similar to each other, and negative or dissimilars pairs. [Hadsell 2006] The policy similarity metric d is transformed to the similarity measure Γ using the Gaussian kernel:

$$\Gamma(x,y) = \exp(-d(x,y)/\beta) \tag{1}$$

For two state spaces and an anchor state y the positive pair is chosen as the one with highest similarity:

$$(\tilde{x}_y, y), \text{where} \tilde{x}_y = \underset{x \in \chi'}{\operatorname{argmax}} \Gamma(x,y). \tag{2}$$

All other states are grouped to negative pairs. Eventually, we aim to map unknown states into this embedding to infer good policies. Therefore, a function is needed that maps states into the embedding. The contrastive loss to learn this mapping is inspired from SimCLR, which also uses self supervised learning and heavy data augmentation to create a representation space of images that can be used for classification. [SimCLR]. The contrastive loss is given by:

$$l_\theta(\tilde{x}_y, y; \chi') = -\log \frac{\Gamma(\tilde{x_y}, y)\exp(\lambda s_\theta(\tilde{x}_y, y))}{\Gamma(\tilde{x}_y, y)\exp(\lambda s_\theta(\tilde{x}_y, y)) + \sum_{x' \in \chi' \setminus \tilde{x}_y} (1-\Gamma(x', y))\exp(\lambda s_\theta(x', y))} \tag{3}$$

This function z_ϕ is composed of an encoding f_ϕ leading to the latent representation of the state and a non-linear projection h_ϕ, following SimCLR. The contrastive loss (Equation 3) is applied to the Projection to train the network. The inference of a suitable policy is done in the representation space as an affine function $\pi_\theta(\cdot \mid y) = W^T f_y + b$ with learnable parameters W and b. The training procedure is depicted in the following Figure.

– Learning figure –

The contrastive metric embeddings learnt with policy metrics are referred to as policy similarity metrics. They are learnt simultaneously during training, where an auxiliary objective derived from the contrastive loss is introduced.

– Algorithm figure –

–Trajectories and stuff –

x_t and y_t are the single states in the optimal trajectories. x_{t+1} is the result of applying the optimal policy π^* to transition dynamics P of the system. $\tilde{x}$ is chosen as the most similar state to the given state y. Then the loss between the Markov processes M_x, M_y are the expected value for y from the optimal trajectory of the contrastive loss (eq 4) of $\tilde{x}$ and y w.r.t. to the optimal trajectory of x. So the loss becomes low, if we can conclude states $\tilde{x}$ that are close to its optimal trajectory from the similarity to y.

3 Experiments from the paper

In this section we want to introduce the paper's main experiment: the jumping task from pixels. The authors also studied generalization on the dm control suite where agents have to ignore visual distraction. However, we want to focus our reproducibility study on the jumping task.

3.1 Jumping Task from Pixels: A Case Study

– Task Figure – n the jumping task, we have an agent that, learning from the image observations on pixel level, needs to jump over an obstacle. The agent can either go right or jump. To avoid a collision the agent needs to time the jump precisely. The environment can be adapted by changing the floor height and the position of the obstacle. Generalization is reached, when the agent becomes invariant to such changes.

– Optimal trajectories figure –

There are 26 different positions of the obstacle and 11 different values of the floor height. To stress generalization, the problem is then split into 18 training tasks and the remaining 268 tasks are evaluation tasks. In the grid below, which corresponds only to the wide grid, each cell corresponds to a different jumping task. The training tasks are marked with the red-letter T.

–Grid figure –

3.2 Classification of the problem

The states of the environment are discrete. This becomes apparent as we are able to classify all possible problems in a grid as combinations of obstacle position and floor height. The actions of the agent are discrete as well: go right or jump. The problem is deterministic because the optimal trajectory can be concluded with certainty. In particular, by processing the fully observable input: all pixel values are available to the agent. Furthermore, the problem is stationary because the optimal trajectory for a given task is time-invariant. As the agent is the only active participant in the environment the problem can be considered a single agent task.
So the task actually falls into the "easiest" category of tasks for reinforcement learning. However, the aim of the authors is to study and improve generalization in RL. And the notion of generalization is very clear in this task and also measurable. Therefore, we think the task is very well chosen to study and show the ability of PSEs to improve generalization in RL.

3.3 Results from the paper

In the paper, three different grid configurations, that represent different styles of generalization are used:

- The wide grid tests generalization via interpolation
- The narrow grid tests out-of-distribution generalization via extrapolation
- The random grid evaluates generalization like supervised learning where the training and test samples are drawn i.i.d. (independently and identically distributed)

All configurations are evaluated using the same hyperparameters that were tuned on the "wide" grid to show the robustness of PSEs to hyperparameter tuning. The authors compare the PSE to other regularization methods. Also the effect of data augmentation using RandConv is studied. The average performance across 100 runs is reported in the following table: