Research Project Proposal:
Transfer of generative models in reinforcement learning

Pierluca D’Oro
pierluca.doro@mail.polimi.it
CSE track
Outline

- Motivation
- State of the art
  - Generative models
  - Model-based RL
- Research idea and plan
• Motivation

• State of the art
  ○ Generative models
  ○ Model-based RL

• Research idea and plan
Reinforcement Learning

An agent acts to maximize a reward collected in an environment.

The RL problem is modeled as a Markov Decision Process:

- States
- Actions
- Initial state distribution
- Reward function
- Discount factor
- \textit{Transition distribution} (i.e., environmental dynamics)

We want the agent to learn the \textit{optimal policy}, possibly estimating the value of a state.
Superhuman Machines?

David et al. *Mastering the game of Go without human knowledge*

Mnih et al. *Playing atari with deep reinforcement learning*
Data collection is hard

For many tasks, collecting experience can be:
- Slow
- Expensive
- Dangerous

In the real world, you cannot speedup time, you have to pay to execute actions or set up a system, and you can break things.

Examples: autonomous driving, robotics, healthcare applications.
Goal of the project

Address data shortage through:
● Sample efficiency
● Transfer from related settings

We plan to leverage:
● Experience generated by **multiple policies**
● Experience generated in **multiple environments**

We want to use it to train a policy for acting in previously unseen scenarios.
Model-based RL

RL approaches can be divided into:
- **Model-free**
- **Model-based**

In model-based RL, the agent uses an approximation of the dynamics of the environment, usually called *model*.

Pros and cons of *model-based RL*:
- ✓ Sample efficiency
- ✓ Easier transfer
- ✗ Bias introduced by the model class

How can we approximate the probability distribution of future states?
• Motivation

• **State of the art**
  ○ **Generative models**
  ○ Model-based RL

• Research idea and plan
Generative Models

They model the distribution that underlies the generation of some data, performing **density estimation**.

Two families of generative models:

- **Explicit** density estimators
  - Modeling the probability density function $p(\mathbf{x})$ of the generating distribution
  - Use simplifying assumptions to maximize data likelihood
  - Examples: autoregressive models, VAE, flow methods

- **Implicit** density estimators
  - Able to draw samples from the approximated distribution
  - Examples: generative adversarial networks
Variational Autoencoders

Hourglass-shaped model by *Kingma and Welling (2013)*:

- An **encoder** $q(z|x)$ maps the input data into latent variables
- A **decoder** $p(x|z)$ converts latent variables into data
- Training for reconstruction of encoded data
- $q(z|x)$ constrained to be as close as possible to a prior distribution $p(z)$
- Samples generated by sampling from the prior $p(z)$ and feeding to $p(x|z)$

A lower bound is maximized in place of the intractable likelihood:

$$
\mathbb{E}_{z \sim q(z|x)} \left[ \log(p(x|z)) \right] - D_{KL} (q(z|x) \mid \mid p(z)) \leq \log(p(x))
$$
Variational Autoencoders
Variational Autoencoders

$p(z) \sim p(x|z)$
Generative Adversarial Networks

Generation is framed as a game:

- A **generator** produces fake samples mimicking a dataset
- A **discriminator** has to distinguish between real and fake samples
- Joint training
- Real and fake samples provided alternately to the discriminator

Original formulation from *Goodfellow et al (2014)*:

\[
\min_G \max_D \mathbb{E}_{x \sim P_{\text{data}}} [\log(D(x))] + \mathbb{E}_{z \sim P(z)} [\log(1 - D(G(z)))]
\]
Generative Adversarial Networks

Latent Space → ~ → Generator

Dataset → ~ → Real Samples → Discriminator

Fake Samples → Generator

Real/Fake prediction → Discriminator
Generative Adversarial Networks

Latent Space → Generator → Fake Samples
How well do they generate?

Wang et al. Video-to-Video Synthesis

Karras et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation

Brock et al. Large Scale GAN Training for High Fidelity Natural Image Synthesis
• Motivation

• **State of the art**
  ○ Generative models
  ○ **Model-based RL**

• Research idea and plan
Modeling single-step transitions

Most of the approaches have modeled the **single-step** transition distribution. Example applications: planning, learning in an “imagined” world, transfer.

Finn and Levine. **Deep visual foresight for planning robot motion**

Ha and Schmidhuber. **Recurrent World Models Facilitate Policy Evolution**

Racanière et al. **Imagination-augmented agents for deep reinforcement learning**
Drawback of single-step modeling

Unrolling single-step future estimates for several timesteps compounds errors.

The reason? *Uncertainty*:
- Error in model estimate
- Environment stochasticity
Modeling trajectories

Co-Reyes et al. **Self-consistent trajectory autoencoder**

Mishra et al. **Prediction and Control with Temporal Segment models**

**Main drawback:** longer trajectories = fewer samples to train our model on!
• **Motivation**

• **State of the art**
  ○ Generative models
  ○ Model-based RL

• **Research idea and plan**
Data is precious

What if the experience is generated by multiple agents acting in a single environment?

What if the experience is generated acting in multiple related environments?

If we want to learn a model to be used by a target policy in a target environment:

- We should not waste data we have at our disposal
- We should consider differences among agent policies and environments
Why it is relevant

Autonomous Driving

Functional Electrical Stimulation
A baseline: the monolithic approach

Consider multiple policies in a fixed environment. We want to learn the **approximate dynamics** of the environment to be used just by the policy that generated the red trajectories.

Using all trajectories:
- ✓ No waste of data
- ✗ Difficult fit (model has limited capacity)

Only using trajectories generated by target policy:
- ✓ Easier fit
- ✗ Waste of data
A baseline: the monolithic approach

Consider multiple policies in a fixed environment. We want to learn the **approximate dynamics** of the environment to be used just by the policy that generated the red trajectories.

Using all trajectories:
- ✓ No waste of data
- ✗ Difficult fit (model has limited capacity)

Only using trajectories generated by target policy:
- ✓ Easier fit
- ✗ Waste of data
Idea: adaptable generative models

Generative models of the dynamics could be adaptable and learn in a clever way from trajectories experienced
- by multiple policies in a single environment,
- in multiple, related, environments.

Desiderata:
- Proper weighting of transitions while learning environmental dynamics
- Faster training or zero-shot transfer for target policies and environments
- Nice theoretical properties (e.g., good efficiency bounds)
Research Plan

Phase 1: Multiple Policies
- Theoretical Analysis
- Experiments
- Analysis of the results
  \textit{First milestone}

Phase 2: Multiple MDPs
- Theoretical Analysis
- Experiments
- Analysis of the results
  \textit{Second milestone}
Thank you for your attention!