# Create batch dataset

The static dataset should be suboptimal, corrupted with exploratory variability (t cover the space to explore, avoid OOD inputs).

In current example projects, the static dataset is collected from one policy of a suboptimal agent: <https://de.mathworks.com/help/reinforcement-learning/ug/train-agent-offline-to-control-furuta-pendulum.html> .

In the case of this thesis, the static dataset derives from different suboptimal explorative runs, randomly chosen.

# Annotate batch dataset with reward

“However, there are some obstacles in applying offline RL to the real world: most algorithms require a reward signal which is often unavailable. To address this issue, we draw inspiration from the literature in inverse RL and reward modelling to learn a reward function.

When the reward signal is not readily available in the environment, it can be learnt. If demonstrations are given, the reward can be learnt either directly with inverse RL [21, 1], or indirectly with generative adversarial imitation learning (GAIL) [13]. If the end goal [8, 32] or reward values [5] are known for a subset of episodes, reward functions could be learnt with supervised learning.

When demonstrations, but no reward signal is available, policies could be learnt with imitation learning [22], and in particular, behavioral cloning (BC) [25, 26]. BC is a supervised learning technique that aims to imitate the exact actions observed in demonstrations.” *2020, Semi-supervised reward learning for offline reinforcement learning, Ksenia Konyushkova*

# Constrained input for control

“For multiple discrete actions, you need to calculate all possible combinations of discrete actions, and use these with rlFiniteSetSpec” <https://it.mathworks.com/matlabcentral/answers/482331-reinforcement-learning-toolbox-multiple-discrete-actions-for-actor-critic-agent-imageinputlayer-is>

“Moreover, this task becomes more demanding, when systems have bounds on the control input, which is referred to as actuator saturation. In the presence of actuator saturation, a given nominal controller may lose its performance or even make the system unstable unless the constraints are considered a priori in the design process.” 2022, *Optimal control of nonlinear systems with inputconstraints using linear time varying approximations Mehmet Itik*

“This example shows how to train a reinforcement learning (RL) agent with actions constrained using the Constraint Enforcement block. This block computes modified control actions that are closest to the actions output by the agent subject to constraints and action bounds.” <https://de.mathworks.com/help/reinforcement-learning/ug/train-reinforcement-learning-agent-with-constraint-enforcement.html>

“Some control applications require the controller to select control actions such that the plant states do not violate certain critical constraints. In many cases, the constraints are on plant states that the controller does not control directly. Instead, you define a constraint function that defines the constraint in terms of the control action signal.”

<https://de.mathworks.com/help/slcontrol/constraint-enforcement.html>

“Therefore, for the applicability of these algorithms, it is important that they execute policies that are guaranteed to perform at least as well as an existing baseline. We can think of the baseline either as a baseline value or the performance of a baseline policy. It is important to note that since the learning algorithms generate these polices from data, they are random variables, and thus, all the guarantees on their performance should be in high probability. This problem has been recently studied under the general title of safety w.r.t. a baseline in bandits and reinforcement learning (RL), in both offline Bottou et al. (2013); Thomas et al. (2015a,b); Swaminathan and Joachims (2015); Petrik et al. (2016) and online Mansour et al. (2015); Wu et al. (2016); Kazerouni et al. (2017); Katariya et al. (2019) settings.

…

Although we are confident that our algorithm will eventually learn a strategy that performs as well as the baseline, and possibly even better, we should control its exploration not to lose too many customers, as a result of providing them with unsatisfactory recommendations.” *2023, Constrained Exploration in Reinforcement Learning Evrard Garcelon*

# Policy Gradient DRL

Algorithms:

1. Distributed Distributional Deterministic Policy Gradients <https://arxiv.org/abs/1804.08617>
2. “Offline RL is an active area of research and many algorithms have been proposed recently, e.g., BCQ [12], MARWIL [33], BAIL [6], ABM [30] AWR [24], and CRR [34]. In our work, we adopt CRR due to its efficiency and simplicity. However, unlike standard offline RL, we do not observe the task rewards.” (see Chapter 2.) *2020, Semi-supervised reward learning for offline reinforcement learning, Ksenia Konyushkova*

Correct input to NN:

<https://it.mathworks.com/matlabcentral/answers/463693-reinforcement-learning-multiple-discrete-actions>